Spring 1-1-2011

Adaptive Formation Flying Maneuvers for Multiple Relative Orbits

Marc Saunders
University of Colorado at Boulder, marc.saunders@colorado.edu

ADAPTIVE FORMATION FLYING MANEUVERS FOR MULTIPLE RELATIVE ORBITS

by

MARC SAUNDERS

B.S., Colorado State University, 1995

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Aerospace Engineering Sciences

2011

This thesis entitled:

Adaptive Formation Flying Maneuvers for Multiple Relative Orbits

written by Marc Saunders

has been approved for the Department of Aerospace Engineering Sciences

_____

Dr. Hanspeter Schaub

_____

Dr. George Born

Date_____

The final copy of this thesis has been examined by the signatories, and we

Find that both the content and the form meet acceptable presentation standards

Of scholarly work in the above mentioned discipline.

Saunders, Marc (M.S., Aerospace)

Adaptive Formation Flying Maneuvers for Multiple Relative Orbits

Thesis directed by Professor Hanspeter Schaub

Abstract:

In order to extend and preserve the mission of an earth orbiting satellite it is imperative that the on board maneuvers do not waste propulsion but maneuver the spacecraft optimally. The challenge for ground stations is to plan maneuvers for spacecraft that will achieve a desired orbit while minimizing fuel costs. Increasing this challenge is the addition of specific keep-out zones (constraints on the spacecraft). For example, a low-earth orbiter (LEO) may need to maintain a specific orbit plane for a sun-synchronous imaging mission but it now has to contend with opposing debris. Computing a maneuver to avoid the debris could have consequences to the mission constraints and cause undesired affects to the desired orbit. The purpose of this research is to develop some techniques that can aid in finding some optimal maneuvers (or maneuvers that use the least amount of energy) and will maintain mission requirements while preserving constraints.

Two different models will be developed that can minimize energy used in the maneuvers. The first model is a linear set of impulsive maneuvers derived from the *Clohessy-Wilshire Equations*. This model can be used as a targeting equation for targeting a specific relative orbit that also minimizes the total energy among a series of maneuvers. The second method is a nonlinear model using a *Lyapunov Function* in a feedback control loop; where the position of a spacecraft relative to a target orbit is minimized and the reference motion can be used to create keep-out zones.

DEDICATION & THANKS

I would like to thank everyone who has supported me in the past couple of years with this endeavor. Without your support and encouragement I would not have been able to make it this far. I would also like to thank Dr. Born for all his advice and taking the time to meet with me. And thanks to Dr. Schaub for pushing me to be better than what I was. This paper would not be possible without them.

Thank you to my wife Joanna and my kids for all their patience, sacrifice and support. And most importantly thanks to God, with whom all things are possible.

CONTENTS

TABLES

FIGURES

CHAPTER 1

INTRODUCTION

1. Motivation

Due to size, power or other constraints when building a satellite; often it is the function of a well equipped ground station that conducts the management of a spacecraft mission. One of the many functions of the ground station is to compute and command station-keeping maneuvers for a vehicle, in order to maintain an orbit that satisfies the mission requirements. Many of these maneuvers can be computed with various techniques. One such technique is to use *Gauss' Variational Equations* [(1)] on the classic orbit elements to compute an impulsive (i.e. instantaneous) delta-velocity maneuver to change a specific orbit element. This technique has the advantage (for example) of maintaining specific orbit parameters like increasing the semi-major axis when atmospheric drag lowers the altitude of a spacecraft. Other analytical maneuver techniques can include (but are not limited too) the classic Hohmann transfer, Bi-elliptic transfer and tangential maneuvers [(2)].

The emphasis in this paper is to investigate a technique to compute maneuvers relative to another orbit. This can be achieved by creating a *Fictitious Orbit* and then compute maneuvers relative to it. For instance, one may design a low-earth sun-synchronous orbit where the semi-major axis and inclination are constant while the ascending node regresses one degree a day due to the oblate equator of the earth. There are no other perturbations on this *Fictitious Orbit* so over time the mean orbit elements [(3)] do not change. Once the desired mission orbit is designed, it is now possible to maneuver a secondary orbit (the actual orbit of the spacecraft) relative to the

primary (fictitious orbit). Since there are many perturbations on the actual orbit, it will drift away from the primary mission designed orbit over time.

Placing the rotating *Hill's Frame* on the primary orbit gives the ability to fly a formation of two spacecraft relative with each other. In this case, it is desirable that the formation consists of one fictional orbit and one actual orbit. Later it will be shown how relative orbit maneuvers can then be used to maintain a mission. This would be accomplished by computing maneuvers relative to the primary orbit that would ultimately change the secondary orbit to match the primary. This technique has an advantage over analytical formulations; where constraints can be added so that maneuvers can attempt to maintain mission requirements while avoiding certain keep-out zones.

2. Potential Applications

This type of maneuvering can be applied to any earth orbiting mission. What is convenient about this application is different types of maneuvers do not need to be distinguished from one another. In many cases, station keeping maneuvers (small correctional maneuvers used to maintain an orbit) are considered different than maneuvers used to dock with a space station. And within station keeping maneuvers, there are many types of maneuvers used to correct different orbit elements depending on the type of orbit and the mission the spacecraft is performing. Computing maneuvers in a relative manner (to another orbit) allows the ground controller the freedom to design a maneuver (no matter the current state of the actual orbit or current time) to achieve what should be the primary mission. For example, certain station keeping maneuvers may only be performed when a spacecraft is eclipsed by the earth or is over a specific target on the ground.

Maneuvering relative to another orbit also has applications when needing to change an orbit completely. If the space shuttle had a mission to conduct maintenance on multiple spacecraft, such as docking with the space station and then leave to repair two other satellites that were in different orbits, this type of maneuvering would be greatly beneficial. All that would need to be done is change the primary orbit in the formulations as the next target and find a series of maneuvers that use the least amount of fuel to get there. If there were obstructions in the path of changing orbits (like debris or sunlight shining on a payload), then these maneuvers can also use some constraint information while still achieving the final orbit.

Finally, when flying a formation of satellites where a secondary deputy satellite has to maintain specific distances relative to a primary satellite (for example remote sensing missions); these maneuvers can be a great advantage. In this case a fictitious primary orbit can be used to maintain the first spacecraft, and then the first spacecraft is then used as the primary target for the second one.

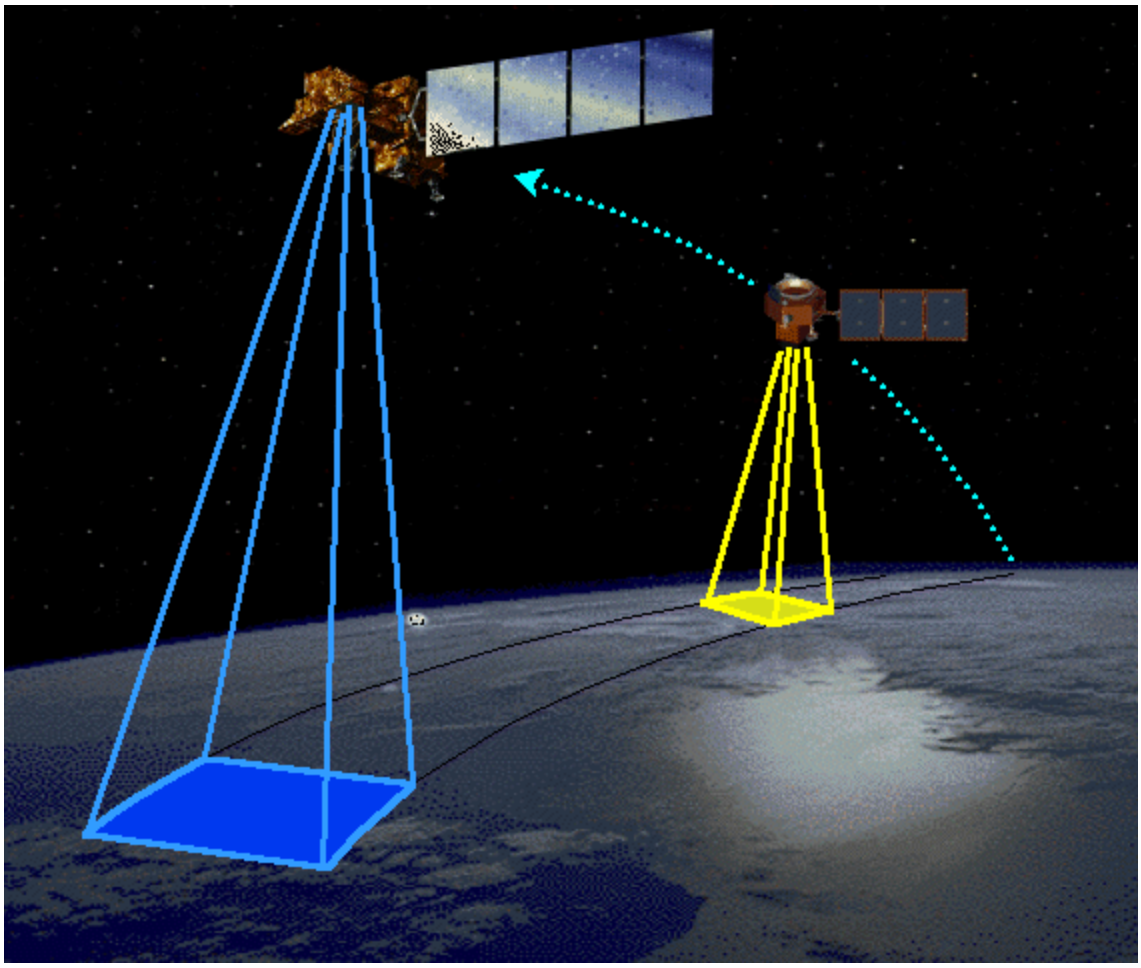

Figure 1: Landsat-7 and EO-1 flying in formation to each other[1].

3. Literature Review

Extensive research had been conducted in the area of formation flying. Much of the literature in this field shows where formation flying equations and techniques in space are derived from. Also, many techniques in regards to this field show how to develop maneuvers

[1] http://eo1.gsfc.nasa.gov/technology/Images/EO-1formationflying.gif

that can maintain a series of formation flying spacecraft together. Among these techniques are developments using mean orbit elements that aid in maintaining *J2* invariant orbits [4].

The techniques used here use traditional methods with Cartesian vectors (no orbit elements). Instead of deriving maneuvers where particular orbit parameters are being maintained like using Gauss' variational equations [3; 5], these maneuvers are computed in reference to using the chief orbit as a fictional ideal orbit that is being maintained.

The techniques of using *least-squares* and *Lyapuov Functions* are primarily used in this text to show how maneuvers can be computed in a relative sense.

CHAPTER 2

TWO-BODY MOTION

1. Equations of Motion

Formation flying or relative motion is the technique of flying one satellite relative to another. To understand how a satellite flies in a formation, it is important to understand how a satellite orbits the earth. Many of the formulations found in this section are used later in order to derive the relative equations of motion. Even though a satellite can fly in formation to another, both are still orbiting the earth. With that in mind, the following sections will show the development of key identities needed for understanding the relative equations of motion. These equations will be needed later to understand how to maneuver one satellite relative to another.

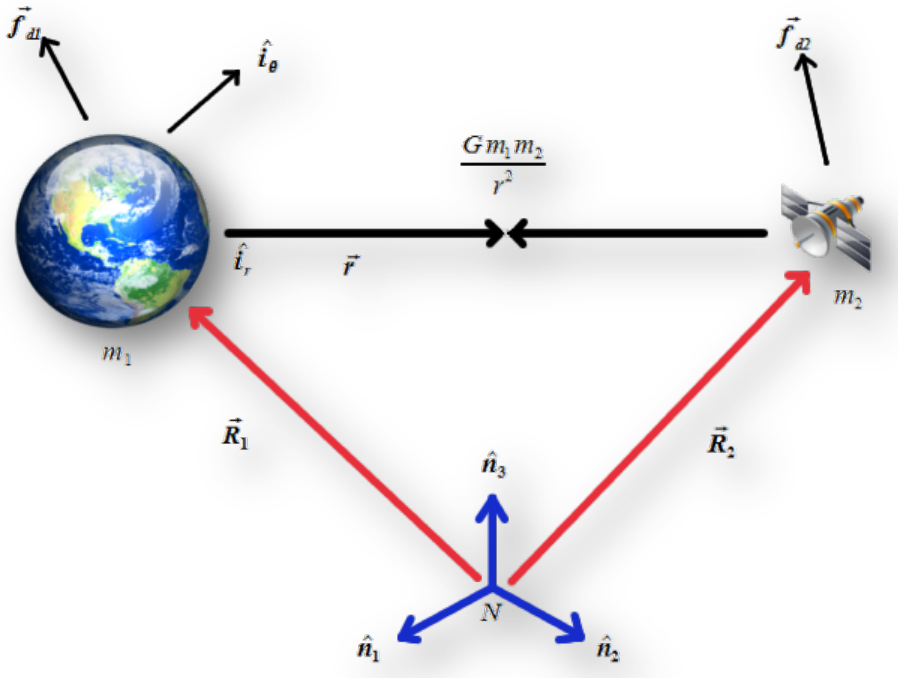

Figure 2: Two-body gravitational motion between the earth and a satellite in inertial space.

To begin, a review of the Two-body equations of motion between an orbiting satellite and the earth are needed. Sir Isaac Newton first introduced his laws of motion in Book I of *The Mathematical Principles of Natural Philosophy* (or simply the *Principa*) [(1)]. There he described the three laws of motion which will be used to develop equations need for later.

Table 1: Newton's Laws [(1)]

Law	**Description**
First	A particle or mass continues in a state of rest or of uniform motion (velocity never changes) in a straight line unless it is compelled to change that state by forces impressed upon it. This law defines that the equations of motion must be in an inertial reference frame. In other words the frame cannot be accelerating.
Second	The rate of change of momentum is proportional to the force impressed and is in the same direction as that force. This is the famous $\boldsymbol{F} = m\boldsymbol{a}$ equation, where $\boldsymbol{F}$ is the vector force, m is the constant mass and **a** is the second-order derivative of the position of the mass (or acceleration).
Third	To every action there is always an opposed equal reaction. This law helps us build a free body diagram as we draw out the forces involved in a system of particles.

In order to derive the equations of motion needed for relative orbits, the equations must be related back to an inertial frame. To do so, a fictional frame (N) is setup somewhere in space that is inertial (not accelerating) $N: \{\hat{\mathbf{n}}_1, \hat{\mathbf{n}}_2, \hat{\mathbf{n}}_3\}$. Also another inertial frame that is fixed to a mass (m_1) $I: \{\hat{\mathbf{i}}_\mathbf{r}, \hat{\mathbf{i}}_\boldsymbol{\theta}, \hat{\mathbf{i}}_\mathbf{h}\}$ is defined. One other important formula that Newton developed in the *Principa* was the law of *Universal Gravitation*. Where he determined that gravity between two point masses (or particles) was the product of their masses and inversely proportional to their distance squared. The force of gravity between the two masses is found by some constant G, also known as the *Universal Gravitational Constant*. In theory any two masses have a

gravitational attraction between each other, but the only reason why it is not noticeable is the constant is extremely small[2]. This value is so small that for all practical applications it is ignored. Although, when dealing with a large mass (like the earth) then the constant cannot be ignored. Note that Eq. (2) is the vector form of Eq. (1).

$$F = \frac{Gm_1 m_2}{r^2} \tag{1}$$

$$\mathbf{F} = \frac{Gm_1 m_2}{r^2}\hat{\mathbf{i}}_r \tag{2}$$

Referring to Figure 1 some definitions can be made in order to find the motion of an earth orbiting satellite. Using vector geometry, it is apparent that the position of the satellite (m_2) with respect to the earth (m_1) can be defined as $\mathbf{R_1} + \mathbf{r} = \mathbf{R}_2$, where the position from the earth to the satellite is defined as $\mathbf{r} = r\hat{\mathbf{i}}_\mathbf{r}$. In this particular case $\mathbf{R}_1$ and $\mathbf{R}_2$ are not matrices but position vectors. Differentiating twice, the motion of the satellite with respect to the earth is defined as

$$\ddot{\mathbf{r}} = \ddot{\mathbf{R}}_2 - \ddot{\mathbf{R}}_1 \tag{3}$$

In order to define the forces involved in this system Newton's second law is used to find the force of gravity. Combining the second law with the *Universal Law of Gravitation* the force of gravity is found (and their disturbance forces $\mathbf{f_{d1}}$ and $\mathbf{f_{d2}}$) on masses m_1 and m_2. The

---

[2] In fact the constant is approximately $6.673e^{-20}$ $km^3/(kg\ s^2)$ [10]

disturbance forces are the sum of all forces that may cause the motion of the satellite to move from the ideal Two-body motion. Such forces can include (but are not limited too) spherical gravity harmonics, third bodies (like the sun and the moon), solar radiation pressure and atmospheric drag.

$$m_1\ddot{\mathbf{R}}_1 = \frac{Gm_1m_2}{r^2}\hat{\mathbf{i}}_r + \mathbf{f}_{d1} \tag{4}$$

$$m_2\ddot{\mathbf{R}}_2 = -\frac{Gm_1m_2}{r^2}\hat{\mathbf{i}}_r + \mathbf{f}_{d2} \tag{5}$$

Solving for the positions of the two masses and substituting them into Eq. (3), a formulation for the motion of the satellite with respect to the earth is found. The disturbance accelerations are defined as $\mathbf{a}_{d1} = \mathbf{f}_{d1}/m_1$, and $\mathbf{a}_{d1} = \mathbf{f}_{d1}/m_1$. The resulting equation of motion is now

$$\ddot{\mathbf{r}} = -\frac{G}{r^2}\hat{\mathbf{i}}_r\left(m_1 + m_2\right) + \mathbf{a}_{d1} + \mathbf{a}_{d2} \tag{6}$$

In order to reduce Eq. (6) some simplifying assumptions can be made. First, assume that the mass of the earth is far greater than the mass of a satellite. So much so, that the mass of the satellite can be neglected $m_1 \gg m_2$ or $m_2 \approx 0$. Also assume that the gravitational force of the earth is far greater than any of the disturbing forces acting on the satellite, as well as the disturbances on the earth are negligible. In other words, the gravity of the earth is the primary force on the satellite $\frac{Gm_1}{r^2} \gg \mathbf{a_{d1}}$, and $\mathbf{a_2} \approx \mathbf{0}$. Combining the *Gravitational Constant* and the

mass of the earth as a single constant the following relationship is defined as $\mu = Gm_1$. Now Eq. (6) can be written into its final reduced form

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} \tag{7}$$

Eq. (7) is the famous equation of motion of an orbiting satellite about the earth in the absence of any disturbances (i.e. Two-body motion). In its current form this ordinary differential equation cannot be solved analytically but only numerically. An analytical solution does exist for Eq. (7) but that is not in the scope of this discussion.

2. Identities

Due to the rotational nature of the *Hill's Frame*, it is necessary to define some relationships to the *angular momentum* of the orbit. Since the relative orbit frame rotates along the *angular momentum* vector the following sections are useful definitions and will be helpful later on when reducing certain derivations.

3. Angular Momentum Constant

From Eq. (7) some useful identities can be derived that will be needed later in our study of relative motion. These identities only work in the Two-body motion case as this assumption is important in order to reduce the equations to simpler and solvable expression. The first identity relates the *angular momentum* of the satellite to the instantaneous change of the true anomaly of the orbit. The *massless angular momentum* is defined as the cross product of the position and velocity of the satellite and is aligned with the third component of the frame I: $\{\hat{\mathbf{i}}_\mathbf{r}, \hat{\mathbf{i}}_\boldsymbol{\theta}, \hat{\mathbf{i}}_\mathbf{h}\}$.

$$\mathbf{h} = \mathbf{r} \times \dot{\mathbf{r}} \tag{8}$$

$$\mathbf{h} = h\hat{\mathbf{i}}_h \tag{9}$$

It can be proven that the *angular momentum* is constant by taking the derivative and showing that it is zero. Again, assuming that any disturbance forces acting on the satellite are negligible; the following is always true

$$\dot{\mathbf{h}} = \dot{\mathbf{r}} \times \dot{\mathbf{r}} + \mathbf{r} \times \ddot{\mathbf{r}} = \mathbf{r} \times \ddot{\mathbf{r}} = \mathbf{r} \times \left(\frac{\mu}{r^3} \mathbf{r} + \mathbf{a}_d \right) = \mathbf{0} \tag{10}$$

Of the six classic orbit elements that can be used to geometrically describe an orbit[3], the true anomaly is the only non-constant value and can be used to describe the instantaneous angular change of an orbit. Taking the derivative of $\mathbf{r} = r\hat{\mathbf{i}}_\mathbf{r}$ with respect to the *I* frame

$$\dot{\mathbf{r}} = \dot{r}\hat{\mathbf{i}}_r + r\dot{f}\hat{\mathbf{i}}_\boldsymbol{\theta} \tag{11}$$

Then substituting the position and velocity terms into the *angular momentum* definition the following can be defined

$$\mathbf{h} = (r\hat{\mathbf{i}}_r) \times (\dot{r}\hat{\mathbf{i}}_r + r\dot{f}\hat{\mathbf{i}}_\theta) = r^2 \dot{f}\hat{\mathbf{i}}_h \tag{12}$$

---

[3] The six classical (or Keplarian) orbital elements are typically $(a, e, i, \Omega, \omega, f)$; semi-major axis, eccentricity, inclination, ascending node, argument of perigee and true anomaly. Most literature interchange's true anomaly with mean anomaly but for our purposes we will be using true anomaly only.

Eq. (12) can be reduced to a scalar form by setting it equal to Eq. (9). This identity shows in a Two-body case that the constant *angular momentum* is equal to the product of the position squared and the instantaneous change of the true anomaly which must also be constant.

$$h = r^2 \dot{f} \tag{13}$$

An identity that relates the second-order derivative of the true anomaly by taking the derivative of Eq. (13) can also be defined. Remembering that the *angular momentum* is constant, and then the following can be true

$$\dot{h} = 0 = 2r\dot{r}\dot{f} + r^2 \ddot{f} \tag{14}$$

Rearranging terms the second-order derivative of the true anomaly is now Eq. (15).

$$\ddot{f} = -2\frac{\dot{r}}{r}\dot{f} \tag{15}$$

4. Angular Momentum and Gravity

Another useful identity is the relationship between the *angular momentum* constant and the gravitational parameter defined in Eq. (7). By taking the derivative of the cross product of the velocity vector and the *angular momentum* vector

$$\frac{d}{dt}(\dot{\mathbf{r}} \times \mathbf{h}) = \ddot{\mathbf{r}} \times \mathbf{h} = -\frac{\mu}{r^3}\mathbf{r} \times (\mathbf{r} \times \dot{\mathbf{r}}) \tag{16}$$

And using the vector triple cross product $\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) \equiv (\mathbf{a} \cdot \mathbf{c})\mathbf{b} - (\mathbf{a} \cdot \mathbf{b})\mathbf{c}$, Eq. (16) can be rewritten into a new form. Using the relationship $\mathbf{r} = r\hat{\mathbf{i}}_\mathbf{r}$ and reducing terms, Eq. (16) can now be written in the following form

$$\ddot{\mathbf{r}} \times \mathbf{h} = \frac{\mu}{r^2}\left(r\dot{\mathbf{r}} - \dot{r}\mathbf{r}\right) \tag{17}$$

Although not obvious, with a little bit of inspection one will notice that Eq. (17) can be rewritten into a new form that allows it to be integrated on both sides of the equation. This is important so that it can be related the gravitational parameter, position and velocity terms only. Look at Eq. (18) carefully, that one can carry out the derivative and get back Eq. (17).

$$\ddot{\mathbf{r}} \times \mathbf{h} = \mu \frac{d}{dt}\left(\frac{\mathbf{r}}{r}\right) \tag{18}$$

$$\frac{d}{dt}\left(\dot{\mathbf{r}} \times \mathbf{h}\right) = \mu \frac{d}{dt}\left(\frac{\mathbf{r}}{r}\right) \tag{19}$$

If both sides of Eq. (19) are integrated; and then solve for a constant of integration then take the dot product of both sides of the equation and use the following dot product identity $\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) \equiv \mathbf{c} \cdot (\mathbf{a} \times \mathbf{b})$ the following is the result.

$$\mathbf{C} = \dot{\mathbf{r}} \times \mathbf{h} - \mu\left(\frac{\mathbf{r}}{r}\right) \tag{20}$$

$$\mathbf{r} \cdot \mathbf{C} = \mathbf{r} \cdot \left(\dot{\mathbf{r}} \times \mathbf{h} - \mu\left(\frac{\mathbf{r}}{r}\right)\right) = h^2 - \mu r \tag{21}$$

$$\mathbf{r} \cdot \mathbf{C} = rC\cos\left(< \mathbf{r}, \mathbf{C}\right) \tag{22}$$

By equating the right sides of Eq. (21) and Eq. (22), the scalar value position can be solved for. This equation is in the form of the polar equation of a conic section [5], Eq. (24).

Geometrically these equations are equivalent so they can be defined as an equation that relates the *angular momentum* to the product of the gravitational parameter and the semi-parameter of a conic section (p).

$$r = \frac{h^2/\mu}{1 + c/\mu \cos(< \mathbf{r}, \mathbf{C})} \tag{23}$$

$$r = \frac{p}{1 + e\cos(f)} \tag{24}$$

$$h^2 = \mu p \tag{25}$$

It is useful as well to find the derivative of Eq. (24) as later it will be shown that the relative equations of motion have the current orbit radius (r) and it's derivative.

$$\dot{r} = \frac{re\dot{f}\sin(f)}{1 + e\cos(f)} \tag{26}$$

CHAPTER 3

FORMATION FLYING

1. Relative Motion

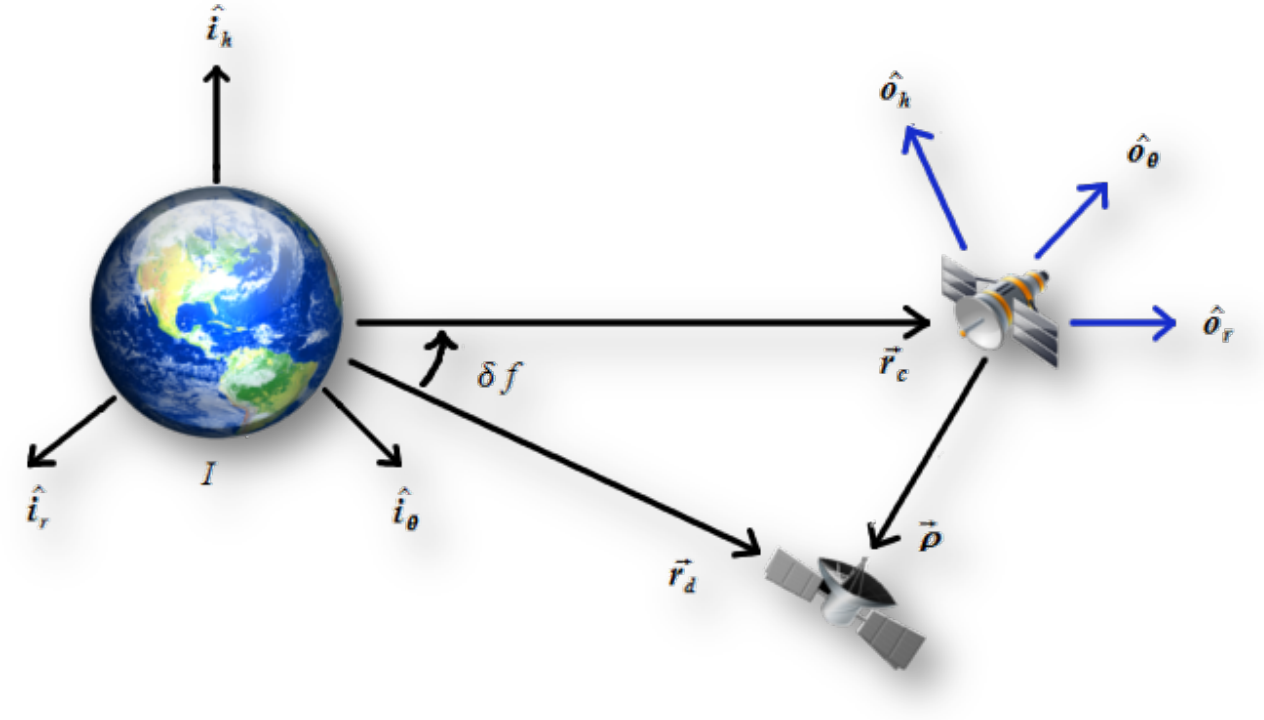

Figure 3: the relative orbit motion between two satellites. For reference, the primary satellite where the rotating frame is defined is called the chief. And the secondary satellite that is flying relative to the chief is the deputy.

Up to this point, only the motion of a single satellite about the earth has been defined. In order to compute maneuvers that are relative to another spacecraft, it would be desirable to be able to relate the motion of one satellite relative to another as they both orbit the earth. An example might be to fly the Space Shuttle relative to the International Space Station when

docking. The following section shows the derivation of the *Clohessy-Wilshire Equations* [6] (in the rotating *Hill's Frame*). These equations are invaluable in order to understand how one might maneuver a spacecraft relative to another. To begin, the full nonlinear, second-order differential equations are derived. After which, some simplifying assumptions are made. Then the equations are reduced to a form where an analytic solution can be found. This solution will help in formulating the first type of maneuvering that can be performed, in order to change the orbit of a deputy spacecraft relative to the chief.

Referring to Figure 3, an inertial frame and a rotating frame for the following derivations ($I:\{\hat{\mathbf{i}}_\mathrm{r}, \hat{\mathbf{i}}_\theta, \hat{\mathbf{i}}_\mathrm{h}\}$ and $O:\{\hat{\mathbf{o}}_\mathrm{r}, \hat{\mathbf{o}}_\theta, \hat{\mathbf{o}}_\mathrm{h}\}$) are defined [5] [7]. The rotating O frame is considered to be the *Hill's Frame* and is centered on the chief orbit. Sometimes this frame is called the local-horizontal/local-vertical frame (LVLH). These two frames are related to each other through the first and third components. The first component relates the two frames via the following identity $\mathbf{r}_\mathrm{c} = r_c\hat{\mathbf{o}}_\mathrm{r}$. Secondly, the two frames are related through the third component lining up with the *angular momentum* vector $\mathbf{h} = h\hat{\mathbf{i}}_\mathrm{h} = h\hat{\mathbf{o}}_\mathrm{h}$. The rotation of the chief orbit and subsequently the rotation of the O frame are found in the relationship $\boldsymbol{\omega}_{O/I} = \dot{f}\hat{\mathbf{o}}_\mathrm{h}$. The relative position of the deputy can be defined in the rotating frame with respect to the chief in Eq. (27) and Eq. (28). [5]

$$\boldsymbol{\rho} = {}^{O}\begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{27}$$

$$\mathbf{r}_d = \mathbf{r}_c + \boldsymbol{\rho} = (r_c + x)\hat{\mathbf{o}}_r + y\hat{\mathbf{o}}_\theta + z\hat{\mathbf{o}}_h \tag{28}$$

The relative velocity (in terms of the rotating frame) of the deputy can be found by taking the derivative of Eq. (28) and using the definitions defined above. Reducing the equation by collecting like terms, the relative velocity is

$$\dot{\mathbf{r}}_d = \left(\dot{r}_c + \dot{x} - y\dot{f}\right)\hat{\mathbf{o}}_r + \left(\dot{y} + r_c\dot{f} + x\dot{f}\right)\hat{\mathbf{o}}_\theta + \dot{z}\hat{\mathbf{o}}_h \tag{29}$$

Similarly, to find the relative acceleration (in terms of the rotating frame), take the derivative of Eq. (29) and collecting like terms, reducing where possible. [5]

$$\ddot{\mathbf{r}}_d = \left(\ddot{r}_c + \ddot{x} - 2\dot{y}\dot{f} - y\ddot{f} - (r_c + x)\dot{f}^2\right)\hat{\mathbf{o}}_r + \left(\ddot{y} + 2(\dot{r}_c + \dot{x})\dot{f} + (r_c + x)\ddot{f} - y\dot{f}^2\right)\hat{\mathbf{o}}_\theta + \ddot{z}\hat{\mathbf{o}}_h \tag{30}$$

Eq. (30) is a second-order nonlinear differential equation that contains second-order terms in it. This would be difficult to solve in this form so it is necessary to reduce it by eliminating the second-order differentials $\ddot{r}_c$ and $\ddot{f}$. Using the following definition $\mathbf{r_c} = r_c\hat{\mathbf{o}}_\mathrm{r}$, and differentiating twice to get the following relationship of the acceleration

$$\ddot{\mathbf{r}}_c = \left(\ddot{r}_c - r_c\dot{f}^2\right)\mathbf{o}_r + \left(2\dot{r}_c\dot{f} + r_c\ddot{f}\right)\mathbf{o}_\theta \tag{31}$$

Making the assumption that the two spacecraft (the deputy and chief) are flying in a Two-body system without disturbances, and using Eq. (7) to make the following relationship, then Eq. (31) can be rewritten.

$$\ddot{\mathbf{r}}_c = -\frac{\mu}{r_c^2}\hat{\mathbf{o}}_r \tag{32}$$

$$\ddot{\mathbf{r}}_c = r_c\dot{f}^2 - \frac{\mu}{r_c^2} \tag{33}$$

Finally, using Eq. (25) and reducing, a relationship for the second-order derivative of the chief orbit radius can be substituted back into Eq. (30). To eliminate the second-order derivative of the true anomaly in Eq. (30), substitute in Eq. (15) to get the final formulation.

$$\ddot{\mathbf{r}}_d = \left(\ddot{x} - 2\left(\dot{y} - y\frac{\dot{r}_c}{r_c} \right)\dot{f} - x\dot{f}^2 - \frac{\mu}{r_c^2} \right)\hat{\mathbf{o}}_r + \left(\ddot{y} + 2\left(\dot{x} - x\frac{\dot{r}_c}{r_c} \right)\dot{f} - y\dot{f}^2 \right)\hat{\mathbf{o}}_\theta + \ddot{z}\hat{\mathbf{o}}_h \tag{34}$$

Earlier the assumption was made that both the deputy and the chief orbits are flying in Two-body motion so now relate Eq. (34) in the rotating frame back to an inertial frame so it can be integrated. With Eq. (28) and using the deputy's x-component in the rotating frame the following definitions can be made.

$$\ddot{\mathbf{r}}_d = -\frac{\mu}{r_d^3}\mathbf{r}_d = -\frac{\mu}{r_d^3}\overset{O}{\begin{pmatrix} r_c + x \\ y \\ z \end{pmatrix}} \tag{35}$$

$$r_d = \sqrt{(r_c + x)^2 + y^2 + z^2} \tag{36}$$

Substituting Eq. (35) into the left-hand-side of Eq. (34) the final nonlinear form of the relative equations of motion are complete. These equations are good for both closed orbit types (circular and elliptical) and they can be inclined as well. The only assumption made up to this point is the dynamics are Two-body so no disturbances are acting on the deputy or the chief orbits.

$$\ddot{x} - 2\left(\dot{y} - y\frac{\dot{r}_c}{r_c}\right)\dot{f} - x\dot{f}^2 - \frac{\mu}{r_c^2} = -\frac{\mu}{r_d^3}(r_c + x) \tag{37a}$$

$$\ddot{y} + 2\left(\dot{x} - x\frac{\dot{r}_c}{r_c}\right)\dot{f} - y\dot{f}^2 = -\frac{\mu}{r_d^3}y \tag{37b}$$

$$\ddot{z} = -\frac{\mu}{r_d^3}z \tag{37c}$$

2. Close Proximity

An advantage of using Eq. (37) is that the deputy and chief orbits can be at any distance from each other, including very large distances. For the purposes of this development, the interest is only in orbits that are in close proximity to one another. It is desirable to be able to maneuver one orbit in order to follow some prescribed path relative to another orbit. Therefore, a second assumption to Eq. (37) can be made to further reduce the formula. By assuming that the components of the vector in Eq. (27) are small compared to the radius of the chief orbit, then Eq. (36) can be rewritten into the following form.

$$r_d = r_c\sqrt{1 + 2\frac{x}{r_c} + \frac{x^2 + y^2 + z^2}{r_c^2}} \tag{38}$$

$$r_d \approx r_c\sqrt{1 + 2\frac{x}{r_c}} \tag{39}$$

Furthermore, Eq. (37) contains terms of r_d^3, which can be modified so that the equation is no longer dependent on the deputy's orbit radius. By using the Taylor Series expansion the

linear form of r_d^3 can be derived by taking the first-order derivative and dropping higher order terms.

$$f(x,y,z) = f(0,0,0) + \frac{\partial f}{\partial x}\Big|_{(0,0,0)} + \frac{\partial f}{\partial y}\Big|_{(0,0,0)} + \frac{\partial f}{\partial z}\Big|_{(0,0,0)} + H.O.T. \tag{40}$$

$$f(x,y,z) = \frac{\mu}{r_d^3} = \frac{\mu}{\left((r_c + x)^2 + y^2 + z^2\right)^{3/2}} \tag{41}$$

$$\frac{\mu}{r_d^3} \approx \frac{\mu}{r_c^3}\left(1 - 3\frac{x}{r_c}\right) \tag{42}$$

Substituting Eq. (42) into Eq. (35) the motion of the deputy in terms of the chiefs orbit radius can now be expressed. Since the current assumption is that the two orbits are in close proximity to one another than a good approximation of the deputy's motion in terms of the chief orbit radius can be defined.

$$-\frac{\mu}{r_d^3}\overset{O}{\begin{pmatrix} r_c + x \\ y \\ z \end{pmatrix}} \approx -\frac{\mu}{r_c^3}\left(1 - 3\frac{x}{r_c}\right)\begin{pmatrix} r_c + x \\ y \\ z \end{pmatrix} \tag{43}$$

The terms in Eq. (43) can be further reduced by carrying out the multiplication of the terms in the $\hat{\mathbf{o}}_\mathrm{r}$ axis and drop the higher order terms based on the current assumption that the components of the relative orbit position are very small compared to the chief orbit radius.

$$\left(1 - 3\frac{x}{r_c}\right)(r_c + x) = r_c + x - 3x - 3\frac{x^2}{r_c} \approx r_c - 2x \tag{44}$$

$$-\frac{\mu}{r_d^3}\overset{O}{\begin{pmatrix} r_c + x \\ y \\ z \end{pmatrix}} \approx -\frac{\mu}{r_c^3}\overset{O}{\begin{pmatrix} r_c - 2x \\ y \\ z \end{pmatrix}} \tag{45}$$

Combining Eq. (13) and Eq. (25) into the $\frac{\mu}{r_c^3}$ term, Eq. (45) can be rewritten into a form that will make it dependent on the change in true anomaly to help reduce Eq. (37).

$$\frac{\mu}{r_c^3} = \frac{r_c \dot{f}^2}{p} \tag{46}$$

Finally, substituting Eq. (45) and Eq. (46) into Eq. (37), combining like terms and reduce. This produces the revised equation of motion for a deputy orbit flying relative to the chief. At this point only two assumptions have been made, that the orbits are flying without disturbances and that they are flying in close proximity to each other. [5]

$$\ddot{x} - 2\dot{f}\left(\dot{y} - y\frac{\dot{r}_c}{r_c}\right) - x\dot{f}^2\left(1 + 2\frac{r_c}{p}\right) = 0 \tag{47a}$$

$$\ddot{y} + 2\dot{f}\left(\dot{x} - x\frac{\dot{r}_c}{r_c}\right) - y\dot{f}^2\left(1 - \frac{r_c}{p}\right) = 0 \tag{47b}$$

$$\ddot{z} + \frac{r_c}{p}\dot{f}^2 z = 0 \tag{47c}$$

3. Linear Solution

At this point it is now beneficial to see if an analytic solution exists for Eq. (47). After careful inspection it is obvious that Eq. (47) is a nonlinear second order differential equation. It carries a dependency on the change in true anomaly which is nonlinear.

In order to find a solution it is necessary to make one more simplifying assumption. Assuming that the chief is flying in a circular orbit (similar to a low-earth or a geo-synchronous orbit's) then the equations of motion can be further reduced. This reduction now makes the second order differential equations linear. Since we are assuming a circular chief orbit, the eccentricity of the orbit plane becomes zero. Consequently, the semi-parameter (p) is now equal to the orbit radius (r_c). Also, the instantaneous change in the orbit radius is now zero, and the change in true anomaly is now equal to the mean motion (n) of the orbit. Eq. (47) reduces to the following

$$\ddot{x} - 2n\dot{y} - xn^2(1+2) = 0 \tag{48a}$$

$$\ddot{y} + 2n\dot{x} - yn^2(1-1) = 0 \tag{48b}$$

$$\ddot{z} = -n^2 z \tag{48c}$$

By simplifying Eq. (48), the final *Clohessy-Wilshire equations* for a relative orbit are found. Since these equations are a linear second-order differential equation, an analytic solution can be found that will yield insight to how a relative orbit will behaves. Later, the solutions to Eq. (48) will be used to begin developing a series of best fit energy impulsive maneuvers. [5]

$$\ddot{x} - 2n\dot{y} - 3n^2 x = 0 \tag{49a}$$

$$\ddot{y} + 2n\dot{x} = 0 \tag{49b}$$

$$\ddot{z} + n^2 z = 0 \tag{49c}$$

To solve Eq. (49), begin by making the observation that the z component is a general spring-mass damper system $a(t) = C_1 \cos(\omega t) + C_2 \sin(\omega t)$ where C_1 and C_2 are constants of integration. Since the z component has no dependency on the x or the y components it is the easiest to solve. For the other two components it will be necessary to solve for one of the components (x or y) first in terms of the other then substitute back into the other components solution. For the z component the general solution can be written as

$$z(t) = C_1 \cos(nt) + C_2 \sin(nt) \tag{50}$$

To find the constants of integration, the solutions are set to some initial conditions of the system. For instance, the initial conditions for the solution will be at $t = 0$.

$$\begin{aligned} x(0) &= x_0 \\ y(0) &= y_0 \\ z(0) &= z_0 \\ \dot{x}(0) &= \dot{x}_0 \\ \dot{y}(0) &= \dot{y}_0 \\ \dot{z}(0) &= \dot{z}_0 \end{aligned} \tag{51}$$

Setting Eq. (50) to $t = 0$, shows that the initial value of the z component is equal to the integration constant C_1. Differentiating Eq. (50) and setting the time to zero again, fins C_2 and the velocity term for z.

$$\dot{z}(t) = \dot{z}_0 \cos(nt) - z_0 n \sin(nt) \tag{52}$$

As stated earlier, to solve for the x and y components, some extra steps must be taken. First, the y component is integrated once to find its velocity term. Just like in the z component

case, the initial conditions are set and the constant of integration is solved for. This yields the following equation for the *y* velocity component.

$$\dot{y}(t) = -2nx + \dot{y}_0 + 2nx_0 \tag{53}$$

Eq. (53) is a function of the *x* component. At this point it is necessary to solve for the *x* component and then substitute that solution back into Eq. (53) to get the final solution for *y*. Substituting Eq. (53) into the *x* component of Eq. (49), and integrating twice by using the general solution (described above) the solution for *x* and its derivative are found. Again, setting the initial conditions in order to solve for the constants of integration will yield the following solutions.

$$x(t) = \frac{\dot{x}_0}{n}\sin(nt) - \left(\frac{2\dot{y}_0}{n} + 3x_0\right)\cos(nt) + \frac{2\dot{y}_0}{n} + 4x_0 \tag{54}$$

$$\dot{x}(t) = \dot{x}_0\cos(nt) + (2\dot{y}_0 + 3nx_0)\sin(nt) \tag{55}$$

Substituting back into Eq. (53), integrating and setting the initial conditions the solutions for the *y* component and its derivative are also found. Together with Eq. (50, 52, 54 and 55) the general solution can be summarized for Eq. (49). [2]

$$x(t) = (4 - 3\cos(nt))x_0 + \frac{\sin(nt)}{n}\dot{x}_0 + \frac{2}{n}(1 - \cos(nt))\dot{y}_0 \tag{56a}$$

$$\dot{x}(t) = 3n\sin(nt)x_0 + \cos(nt)\dot{x}_0 + 2\sin(nt)\dot{y}_0 \tag{56b}$$

$$y(t) = 6(\sin(nt) - nt)x_0 + y_0 + \frac{2}{n}(\cos(nt) - 1)\dot{x}_0 + \left(\frac{4}{n}\sin(nt) - 3t\right)\dot{y}_0 \tag{56c}$$

$$\dot{y}(t) = 6n(\cos(nt) - 1)x_0 - 2\sin(nt)\dot{x}_0 + (4\cos(nt) - 3)\dot{y}_0 \tag{56d}$$

$$z(t) = \cos(nt)z_0 + \frac{\sin(nt)}{n}\dot{z}_0 \tag{56e}$$

$$\dot{z}(t) = -n\sin(nt)z_0 + \cos(nt)\dot{z}_0 \tag{56f}$$

It is interesting to note from Eq. (56) that the deputy orbit will maintain a closed loop about the chief, so long as there is no secular drift that occurs. Inspecting the solution further, the only possible secular drift that could happen is when $\dot{y}_0$ starts off as non-zero. For example, setting up the following initial conditions for a geo-synchronous circular orbit and updating for one period[4], we should see a closed loop for the deputy's motion. This statement is shown to be the case in Figure 4, where a closed loop relative orbit is achieved using the following initial conditions.

$$\boldsymbol{\rho}_0 = \begin{pmatrix} -0.1 \\ 0.2 \\ -0.2 \\ 0.01 \\ 0 \\ -0.02 \end{pmatrix} [m, m/s]$$

[4] One sidereal day is approximately 1.002737909350795 [(2)].

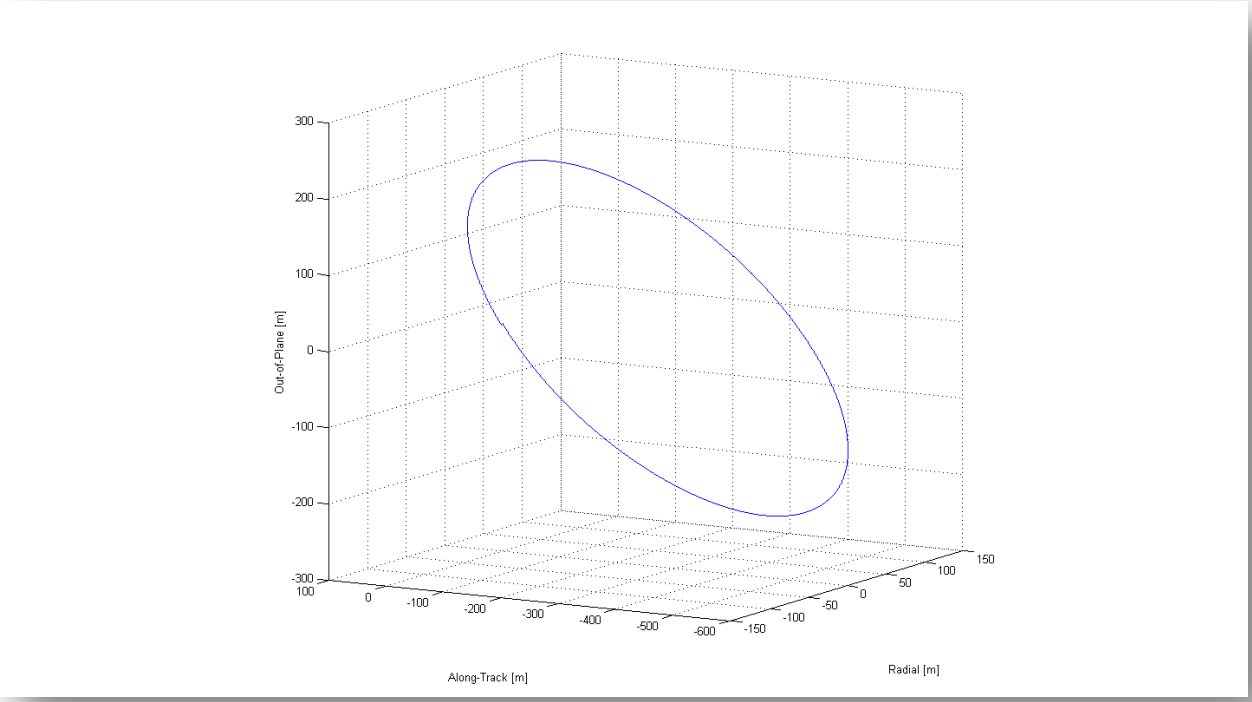

Figure 4: an example of a closed relative orbit. With no secular drift, the deputy will continue to loop around the chief.

When using the same initial conditions as the previous example and setting the initial condition of $\dot{y}_0$ to *0.001m/s*, it becomes apparent that the deputy will drift apart from the chief almost immediately, as in Figure 5.

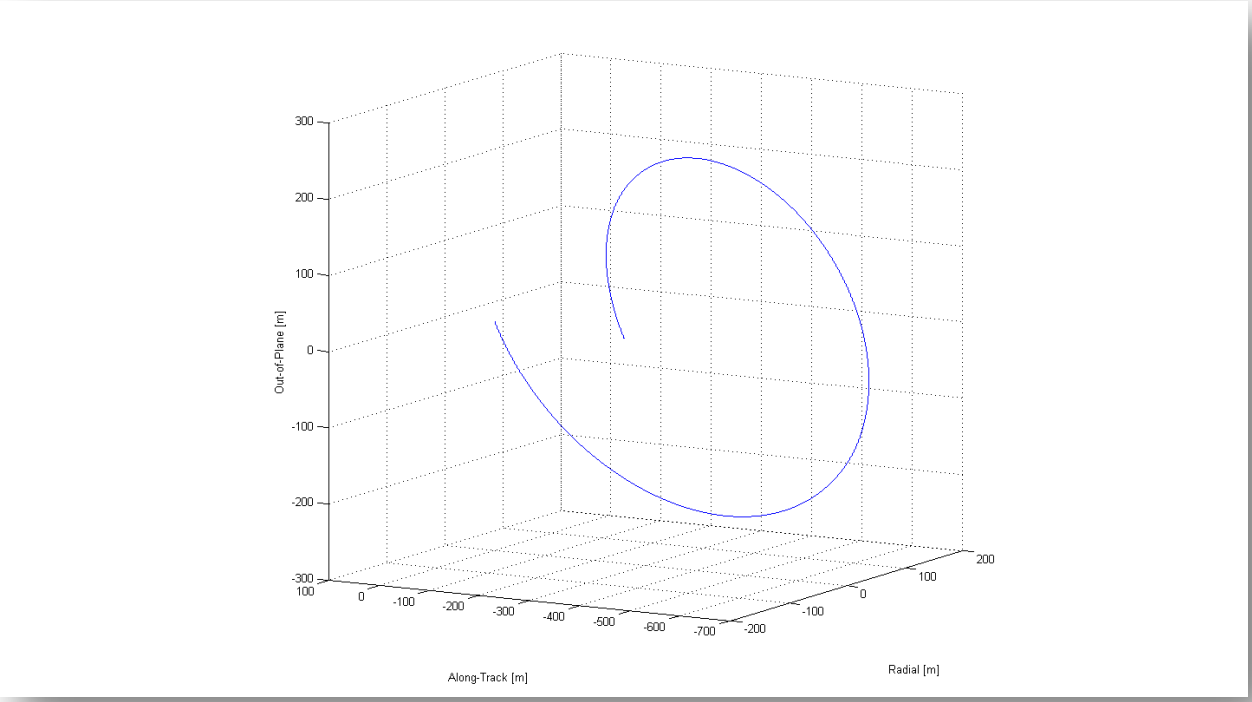

Figure 5: an example of an un-closed relative orbit. With secular drift, the deputy will drift away.

4. Inertial to Rotating Frames and Back

It is often necessary at times to be able to convert from the inertial frame to the *Hill's* rotating frame and back again. Remember that the first assumption was that both the chief and deputy orbits fly in the absence of any disturbances. Then the following definitions also allow one to compute the acceleration in the rotating frame with disturbances.

These definitions help in converting relative states back to the inertial, so that one could (for example) see how the deputy's orbit changes over time. In order to take advantage of these equations with an equation of motion that includes disturbances, one would need to numerically integrate both the deputy and the chief in the inertial frame first. Then using the numerically

solved solutions they can then be used to compute the instantaneous relative position, velocity and accelerations. Keep in mind by doing this there will be drift in the relative position even when the initial condition of $\dot{y}_0$ is set to zero.

To convert from the inertial Cartesian vectors to the rotating relative frame start with Eq. (27) and (28). To find the rotating velocity, differentiate once then differentiate twice to find the acceleration.

$$\boldsymbol{\rho} = \mathbf{ON}\left(\mathbf{r}_d - \mathbf{r}_c\right) \tag{57a}$$

$$\dot{\boldsymbol{\rho}} = \mathbf{ON}\left(\dot{\mathbf{r}}_d - \dot{\mathbf{r}}_c\right) - \dot{f}\left(x\hat{\mathbf{o}}_\theta - y\hat{\mathbf{o}}_r\right) \tag{57b}$$

$$\ddot{\boldsymbol{\rho}} = \mathbf{ON}\left(\ddot{\mathbf{r}}_d - \ddot{\mathbf{r}}_c\right) - \dot{f}\left(\left(x - \dot{f}y\right)\hat{\mathbf{o}}_\theta + \left(\dot{f}x - y\right)\hat{\mathbf{o}}_r\right) \tag{57c}$$

To convert from the rotating vectors back to the inertial frame use the following equations.

$$\mathbf{r}_d = \mathbf{r}_c + \mathbf{NO}\boldsymbol{\rho} \tag{58a}$$

$$\dot{\mathbf{r}}_d = \dot{\mathbf{r}}_c + \mathbf{NO}\left(\dot{\boldsymbol{\rho}} + \dot{f}\left(x\hat{\mathbf{o}}_\theta - y\hat{\mathbf{o}}_r\right)\right) \tag{58b}$$

$$\ddot{\mathbf{r}}_d = \ddot{\mathbf{r}}_c + \mathbf{NO}\left(\ddot{\boldsymbol{\rho}} + \dot{f}\left(\left(x - \dot{f}y\right)\hat{\mathbf{o}}_\theta + \left(\dot{f}x - y\right)\hat{\mathbf{o}}_r\right)\right) \tag{58c}$$

The rotation matrix from the inertial *N* frame to the rotating *O* frame is defined as

$$\mathbf{ON} = \begin{pmatrix} \hat{\mathbf{o}}_r^T \\ \hat{\mathbf{o}}_\theta^T \\ \hat{\mathbf{o}}_h^T \end{pmatrix} \tag{59}$$

And the transpose of Eq. (59) is from *O* to *N* frames are

$$\mathbf{NO} = \mathbf{ON}^T \tag{60}$$

CHAPTER 4

STATE TRANSITION MATRIX

Now that an analytical solution for a relative orbit has been developed, it is possible to take advantage of the solution in a linear system of equations. Keeping in mind the assumptions until now, Two-body motion, flying in close proximity and using only circular orbits a state transition matrix can be defined that can update a relative orbit state vector from an initial time to a final time. Understanding how a linear set of equations behaves with this motion is important to develop a set of maneuvers that can be used to update a relative orbit.

1. Circular

To begin the development of a linear system (and to define the state transition matrix) it is first necessary to define the state vector. In this case it is the combination of the relative position and velocity.

$$\mathbf{X} = \begin{pmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} \tag{61}$$

With Eq. (56), the terms can be arranged properly to write the state transition matrix. This matrix can then be used to update the relative position (and velocity) over time as long as the chief orbit is circular. [8]

$$\mathbf{\Phi} = \begin{bmatrix} 4-3\cos(nt) & 0 & 0 & \frac{\sin(nt)}{n} & \frac{2}{n}\left(1-\cos(nt)\right) & 0 \\ 6\left(\sin(nt)-nt\right) & 1 & 0 & \frac{2}{n}\left(\cos(nt)-1\right) & \frac{4}{n}\sin(nt)-3t & 0 \\ 0 & 0 & \cos(nt) & 0 & 0 & \frac{\sin(nt)}{n} \\ 3n\sin(nt) & 0 & 0 & \cos(nt) & 2\sin(nt) & 0 \\ 6n\left(\cos(nt)-1\right) & 0 & 0 & -2\sin(nt) & 4\cos(nt)-3 & 0 \\ 0 & 0 & -n\sin(nt) & 0 & 0 & \cos(nt) \end{bmatrix} \tag{62}$$

2. Elliptical

Although Eq. (62) works only for circular orbits, a state transition matrix can still be solved numerically for elliptic orbits. It is possible to find an analytic solution for elliptic orbits by using orbit element differences [7]. For this discussion, a solution can be found in Cartesian space. This is accomplished by using numerical integration [9], by initializing the state transition matrix to the identity then the general solution from Eq. (63) can be solved.

$$\mathbf{X}(t) = \Phi(t, t_0)\mathbf{X}_0 \tag{63}$$

As mentioned before, to solve for the state transition matrix in the elliptical case it is possible to numerically integrate the following expression, where the **A** matrix is the partial differentials for the system of motion [8].

$$\dot{\mathbf{\Phi}}(t, t_0) = \mathbf{A}(t)\mathbf{\Phi}(t, t_0) \tag{64}$$

To define the **A** matrix, use Eq. (47) and take the partials with respect to each other. Now using the state vector Eq. (61) take the first derivative to find the following

$$\dot{\mathbf{X}} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ 2\dot{f}\left(\dot{y} + y\dfrac{\dot{r}_c}{r_c}\right) + x\dot{f}^2\left(1 + 2\dfrac{r_c}{p}\right) \\ -2\dot{f}\left(\dot{x} - x\dfrac{\dot{r}_c}{r_c}\right) + y\dot{f}^2\left(1 - \dfrac{r_c}{p}\right) \\ -\dfrac{r_c}{p}\dot{f}^2 z \end{pmatrix} \tag{65}$$

The **A** matrix is defined as the partials of the derivative of **X** with respect to **X** itself. Similar to techniques used in orbit determination [8] the **A** matrix is shown in Eq. (66) and (67).

$$\mathbf{A} = \frac{\partial \dot{\mathbf{X}}}{\partial \mathbf{X}} = \begin{bmatrix} \partial\dot{x}/\partial x & \partial\dot{x}/\partial y & \partial\dot{x}/\partial z & \partial\dot{x}/\partial\dot{x} & \partial\dot{x}/\partial\dot{y} & \partial\dot{x}/\partial\dot{z} \\ \partial\dot{y}/\partial x & \partial\dot{y}/\partial y & \partial\dot{y}/\partial z & \partial\dot{y}/\partial\dot{x} & \partial\dot{y}/\partial\dot{y} & \partial\dot{y}/\partial\dot{z} \\ \partial\dot{z}/\partial x & \partial\dot{z}/\partial y & \partial\dot{z}/\partial z & \partial\dot{z}/\partial\dot{x} & \partial\dot{z}/\partial\dot{y} & \partial\dot{z}/\partial\dot{z} \\ \partial\ddot{x}/\partial x & \partial\ddot{x}/\partial y & \partial\ddot{x}/\partial z & \partial\ddot{x}/\partial\dot{x} & \partial\ddot{x}/\partial\dot{y} & \partial\ddot{x}/\partial\dot{z} \\ \partial\ddot{y}/\partial x & \partial\ddot{y}/\partial y & \partial\ddot{y}/\partial z & \partial\ddot{y}/\partial\dot{x} & \partial\ddot{y}/\partial\dot{y} & \partial\ddot{y}/\partial\dot{z} \\ \partial\ddot{z}/\partial x & \partial\ddot{z}/\partial y & \partial\ddot{z}/\partial z & \partial\ddot{z}/\partial\dot{x} & \partial\ddot{z}/\partial\dot{y} & \partial\ddot{z}/\partial\dot{z} \end{bmatrix} \tag{66}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \dot{f}^2\left(1 + 2\dfrac{r_c}{p}\right) & 2\dot{f}\dfrac{\dot{r}_c}{r_c} & 0 & 0 & 2\dot{f} & 0 \\ 2\dot{f}\dfrac{\dot{r}_c}{r_c} & -\dot{f}^2\left(\dfrac{r_c}{p} - 1\right) & 0 & -2\dot{f} & 0 & 0 \\ 0 & 0 & -\dot{f}^2\dfrac{r_c}{p} & 0 & 0 & 0 \end{bmatrix} \tag{67}$$

CHAPTER 5

BEST FIT MANEUVERS

When flying a deputy satellite relative to the chief, it is now possible to "move" or change its position according to the motion of the chief. Other than just changing the deputies' position for formation flying (like docking with another satellite) it is possible to define the chief as a fictitious orbit. In other words, conduct a formation flying mission where an actual spacecraft flies relative to an ideal orbit. By defining the chief as a fictitious orbit it can be used as the mission requirements orbit. Meaning, in the absence of disturbances that may cause the satellite to move out of its planned motion (i.e. atmospheric drag or solar radiation pressure) the chief orbit will always maintain the mission requirements. The deputy is now defined as the "actual" satellite's orbit with disturbances. Over time as the chief maintains the ideal orbit, the deputy will drift from the ideal. Using the following derivations, it is now possible to maneuver the deputy back to the ideal orbit using a series of best fit maneuvers. These impulsive maneuvers are instant changes to the velocity of the deputy. In reality a ground station will use the spacecrafts thruster models to convert these maneuvers to some amount of burn time for the thruster's on board the vehicle.

1. Targeting Equation

A targeting equation is a form of using the initial conditions (position and velocity) of a satellite and some prescribed linear motion to achieve the ending condition. By using the state transition matrix, an impulsive maneuver can be found that will help achieve the end result given the initial and final conditions. To begin this derivation, start by rewriting Eq. (63) into a vector form and use the following definitions.

$$\begin{pmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{pmatrix} = \begin{bmatrix} \boldsymbol{\varphi}_1 & \boldsymbol{\varphi}_2 \\ \boldsymbol{\varphi}_3 & \boldsymbol{\varphi}_4 \end{bmatrix} \begin{pmatrix} \mathbf{x}_0 \\ \dot{\mathbf{x}}_0 \end{pmatrix} \tag{68}$$

$$\mathbf{X} = \begin{pmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{pmatrix} \text{ And } \boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\varphi}_1 & \boldsymbol{\varphi}_2 \\ \boldsymbol{\varphi}_3 & \boldsymbol{\varphi}_4 \end{bmatrix} \tag{69}$$

Introducing a change in velocity to the initial velocity component into Eq. (68) and solving for the delta, two types of targeting equations can be defined. One equation is defined for achieving a desired final position. The other equation is defined for achieving a desired final velocity.

$$\begin{pmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{pmatrix} = \begin{bmatrix} \boldsymbol{\varphi}_1 & \boldsymbol{\varphi}_2 \\ \boldsymbol{\varphi}_3 & \boldsymbol{\varphi}_4 \end{bmatrix} \begin{pmatrix} \mathbf{x}_0 \\ \dot{\mathbf{x}}_0 + \Delta\mathbf{v} \end{pmatrix} \tag{70}$$

$$\Delta\mathbf{v}_a = \boldsymbol{\varphi}_2^{-1}\left(\mathbf{x} - \boldsymbol{\varphi}_1\mathbf{x}_0\right) - \dot{\mathbf{x}}_0 \tag{71}$$

$$\Delta\mathbf{v}_b = \boldsymbol{\varphi}_4^{-1}\left(\dot{\mathbf{x}}_1 - \boldsymbol{\varphi}_3\mathbf{x}_0\right) - \dot{\mathbf{x}}_0 \tag{72}$$

2. Targeting Example

Take for example, an initial relative state of the deputy orbit (at GEO[5]) with some relative drift from the chief and an offset in the position. Using the targeting equations Eq. (71) and (72), find two maneuvers set apart by ten minutes each to achieve a new relative state.

---

[5] A geo-synchronous orbit (GEO) has the same mean motion as the earth at $7.29211585529998e-5\ \frac{\text{rad}}{\text{s}}$

$$\mathbf{X}_0 = \begin{pmatrix} -50 \\ 200 \\ -75 \\ 0.1 \\ 0.01 \\ -0.02 \end{pmatrix} [m, m/s]$$

The final (desired) relative state is to perch (zero relative velocity) the deputy one-hundred meters to the west of the chief.

$$\mathbf{X}_f = \begin{pmatrix} 0 \\ -100 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} [m, m/s]$$

The state transition matrix for updating the deputy for both maneuvers is the same since they are both ten minutes apart. Using Eq. (62) the STM is

$$\mathbf{\Phi}(600,0) = \begin{bmatrix} 1.0029 & 0 & 0 & 599.81 & 26.247 & 0 \\ -8.3748e-005 & 1 & 0 & -26.247 & 599.23 & 0 \\ 0 & 0 & 0.99904 & 0 & 0 & 599.81 \\ 9.5684e-006 & 0 & 0 & 0.99904 & 0.087477 & 0 \\ -4.1871e-007 & 0 & 0 & -0.087477 & 0.99617 & 0 \\ 0 & 0 & -3.1895e-006 & 0 & 0 & 0.99904 \end{bmatrix}$$

Both maneuvers are on the order of about five-hundred centimeters per second, and when applied at the correct times the final state is achieved. Keep in mind that it is important that after

the first maneuver is applied then the second maneuver's initial state is now the final state after first maneuver has applied.

$$\Delta v_a = \begin{pmatrix} 0.005 & -0.506 & 0.145 \end{pmatrix} \frac{m}{s}$$

$$\Delta v_b = \begin{pmatrix} -0.061 & 0.503 & -0.125 \end{pmatrix} \frac{m}{s}$$

3. Least Squares Solution

The targeting equation is a great method for computing a single impulsive maneuver (or a pair of maneuvers), but it cannot be used for multiple maneuvers to target a final state. Also, when applying the first maneuver, it does not have knowledge of where the final state will be when the next maneuver is applied. This is because the single maneuver can only change three degrees of freedom and the relative orbit state has six. That is why in the previous example it required two maneuvers to achieve the final relative state. And even then, when the second maneuver is applied some time has passed with the relative state has changed.

If multiple maneuvers (more than two) are conducted in a series (for example every ten minutes) until a final orbit is achieved, then it would be possible for all the maneuvers to be computed in a best fit case. With this idea in mind, it would be possible to extend Eq. (70) to allow multiple maneuvers to occur to achieve a final relative state. In order to achieve this, it is possible to take advantage of the linear nature of the system of linear equations.

Since it takes exactly two maneuvers to achieve the final relative state, then computing more maneuvers would make this an over-determined system (more equations than unknowns)

[8]. To solve this equation, the *performance index* **J** can be used [8]. In an over-determined system it is desirable to reduce the amount of error by using the *performance index*.

$$J = \frac{1}{2}\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon} \tag{73}$$

In Eq. (73), the epsilon term is the amount of error in a linear system of equations. In this case for relative motion, the error in the system is the difference between the states of the chief orbit versus the deputy. Keeping in mind if there is an initial in-track velocity component in the deputy then there will be a difference (error) in orbit states over time (they will drift apart). The attempt is being made to correct the error between the chief and deputy orbits by a series of best fit maneuvers. In order to find a set of maneuvers that changes the state of the deputy and move it to the chief, rewrite Eq. (63) into the following while accounting for some error (epsilon) to be between the chief and the deputy states.

$$\mathbf{X} = \boldsymbol{\Gamma}\mathbf{X}_0 + \boldsymbol{\varepsilon} \tag{74}$$

Eq. (74) is considered to be a series of state transitions. At each step in time, the initial state can be multiplied by a state transition matrix to find the final state. The vector form of Eq. (74) is

$$\begin{aligned} \mathbf{X}_1 &= \boldsymbol{\Phi}(t_1, t_0)\mathbf{X}_0 + \boldsymbol{\varepsilon}_1 \\ \mathbf{X}_2 &= \boldsymbol{\Phi}(t_2, t_1)\mathbf{X}_1 + \boldsymbol{\varepsilon}_2 \\ &\vdots \\ \mathbf{X}_n &= \boldsymbol{\Phi}(t_n, t_{n-1})\mathbf{X}_{n-1} + \boldsymbol{\varepsilon}_n \end{aligned} \tag{75a}$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_n \end{bmatrix}; \quad \mathbf{\Gamma} = \begin{bmatrix} \mathbf{\Phi}(t_1, t_0) \\ \vdots \\ \mathbf{\Phi}(t_n, t_{n-1}) \end{bmatrix}; \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \boldsymbol{\varepsilon}_1 \\ \vdots \\ \boldsymbol{\varepsilon}_n \end{bmatrix} \tag{75b}$$

To reduce the errors (or differences between states) in the system, it is desirable to have the partials of the *performance index* with respect to the state vector be zero $\left(\frac{\partial J}{\partial \boldsymbol{X}} = 0\right)$. Substituting Eq. (74) into Eq. (73), taking the partials and setting it to zero results in the following

$$-\mathbf{\Gamma}^T\left(\mathbf{X} - \mathbf{\Gamma}\mathbf{X}_0\right) = 0 \tag{76}$$

When solving for the final state vector (**X**), this will produce the state that minimizes all the errors in the system over time.

$$\mathbf{X} = \left(\mathbf{\Gamma}^T\mathbf{\Gamma}\right)^{-1}\mathbf{\Gamma}^T\mathbf{X}_0 \tag{77a}$$

$$\mathbf{X} = \left(\mathbf{\Gamma}^T\mathbf{W}\mathbf{\Gamma}\right)^{-1}\mathbf{\Gamma}^T\mathbf{W}\mathbf{X}_0 \tag{77b}$$

Eq. (77) is in the same form as the classic least-squares and weighted least-squares solution [8] [10]. Keep in mind, that computing the inverse of the matrix will only be valid if $\mathbf{\Gamma}^T\mathbf{\Gamma}$ is positive definite. In other words, the linear system of equations must be linearly independent from one another.

Now use Eq. (77) and solve for a series of maneuvers to move the deputy orbit to the fictitious ideal chief orbit that maintains the mission requirements. The first step is to define the total difference between the chief and deputy orbits in the system. This is established by

defining an *aim-point* in the orbit. An *aim-point* could be any final state in the orbit where the actual orbit should be at some future time. Looking ahead at some future time it would be possible to see what the state of the ideal orbit would be and define that as the *aim-point*. Therefore, at time (t_f) the ideal chief orbit is known and will be at some state $\mathbf{X}_f$. Since the initial position is known for the actual orbit, then the actual orbit can be updated to the final time as well; and taking the difference between the actual and ideal orbits, will yield the total difference of the system.

$$\boldsymbol{\delta}_T = \mathbf{X}_f - \boldsymbol{\Phi}\left(t_f, t_0\right)\mathbf{X}_0 \tag{78}$$

Upon inspecting Eq. (56) and Eq. (62), one will notice that the second and fourth quadrants of Eq. (69) are the only quadrants of the state transition matrix that change the velocity of the state vector. These quadrants are only used because solving for the velocity component of the state vector will allow for creating maneuvers that will change the state in the system. Given *n* number of maneuvers to plan, then a series of state transitions for the velocity component can be concatenated into a *6*x*3n* matrix.

$$\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{\varphi}_2\left(t_f, t_0\right) & \boldsymbol{\varphi}_2\left(t_f, t_1\right) & \boldsymbol{\varphi}_2\left(t_f, t_2\right) & \cdots & \boldsymbol{\varphi}_2\left(t_f, t_{n-1}\right) \\ \boldsymbol{\varphi}_4\left(t_f, t_0\right) & \boldsymbol{\varphi}_4\left(t_f, t_1\right) & \boldsymbol{\varphi}_4\left(t_f, t_2\right) & \cdots & \boldsymbol{\varphi}_4\left(t_f, t_{n-1}\right) \end{bmatrix} \tag{79}$$

In a least-squares sense, this is the set of linearly independent equations (or *observations*) of the system. The variance-covariance matrix for the non-weighted and weighted system would be

$$\mathbf{P} = \left(\boldsymbol{\Psi}^T \boldsymbol{\Psi}\right)^{-1} \tag{80a}$$

$$\mathbf{P} = \left(\mathbf{\Psi}^T \mathbf{W} \mathbf{\Psi}\right)^{-1} \tag{80b}$$

The set of impulsive maneuvers would be the solution to the system of linear equations. Beginning with Eq. (77), it is now possible to setup a similar formulation using Eq. (79). Instead of using a series of state transitions matrices, use the series of state transitions of only the velocity components. With the following formulation $\Delta \boldsymbol{v} = (\mathbf{\Psi}^T \mathbf{\Psi}) \mathbf{\Psi}^T \boldsymbol{\delta}_T$ and substituting in Eq. (80), the maneuvers can be defined as

$$\Delta \mathbf{v} = \mathbf{P} \mathbf{\Psi}^T \boldsymbol{\delta}_T \tag{81a}$$

$$\Delta \mathbf{v} = \mathbf{P} \mathbf{\Psi}^T \mathbf{W} \boldsymbol{\delta}_T \tag{81b}$$

The resulting set of impulsive maneuvers is the set of maneuvers that best fits the system of change between where the final actual orbit and ideal orbit would be. Take note, that the result is a vector of *3n*x*1* elements. If this vector is reshaped into a matrix of *3*x*n* then the final set of maneuvers would be found as a set of column vectors. Consequently, this best fit for the maneuvers is the best estimate for reducing the amount of error (or difference) between the chief and the deputy ($\boldsymbol{\varepsilon} = \boldsymbol{\delta}_T - \mathbf{\Psi} \Delta \boldsymbol{v}$).

For example, start with an initial relative state for the deputy to be the following

$$\mathbf{X}_0 = \begin{pmatrix} 150 \\ -3000 \\ 200 \\ -0.3 \\ 0.02 \\ -0.01 \end{pmatrix} [m, m/s]$$

Assume the chief is in a geo-synchronous circular orbit and it's desired to use as little fuel as possible to move the deputy into a parking orbit at one-hundred meters trailing the chief with a small drift rate away from the chief.

$$\mathbf{X}_f = \begin{pmatrix} 0 \\ 100 \\ 0 \\ 0 \\ 0.01 \\ 0 \end{pmatrix} [m, m/s]$$

In this case, perform six maneuvers every ten minutes for the next hour to achieve the final parking orbit. Using Eq. (81) (without any weighting), compute the following impulsive maneuvers

Table 2: Relative orbit impulsive maneuvers

Time [s]	Maneuver Magnitude [m/s]	Impulsive Maneuver [m/s]
0	0.712	[0.004, 0.711, -0.042]
600	0.425	[0.026, 0.423, -0.024]
1200	0.137	[0.023, 0.135, -0.007]
1800	0.153	[-0.006, -0.152, 0.011]
2400	0.441	[-0.059, -0.436, 0.028]
3000	0.728	[-0.138, -0.713, 0.046]

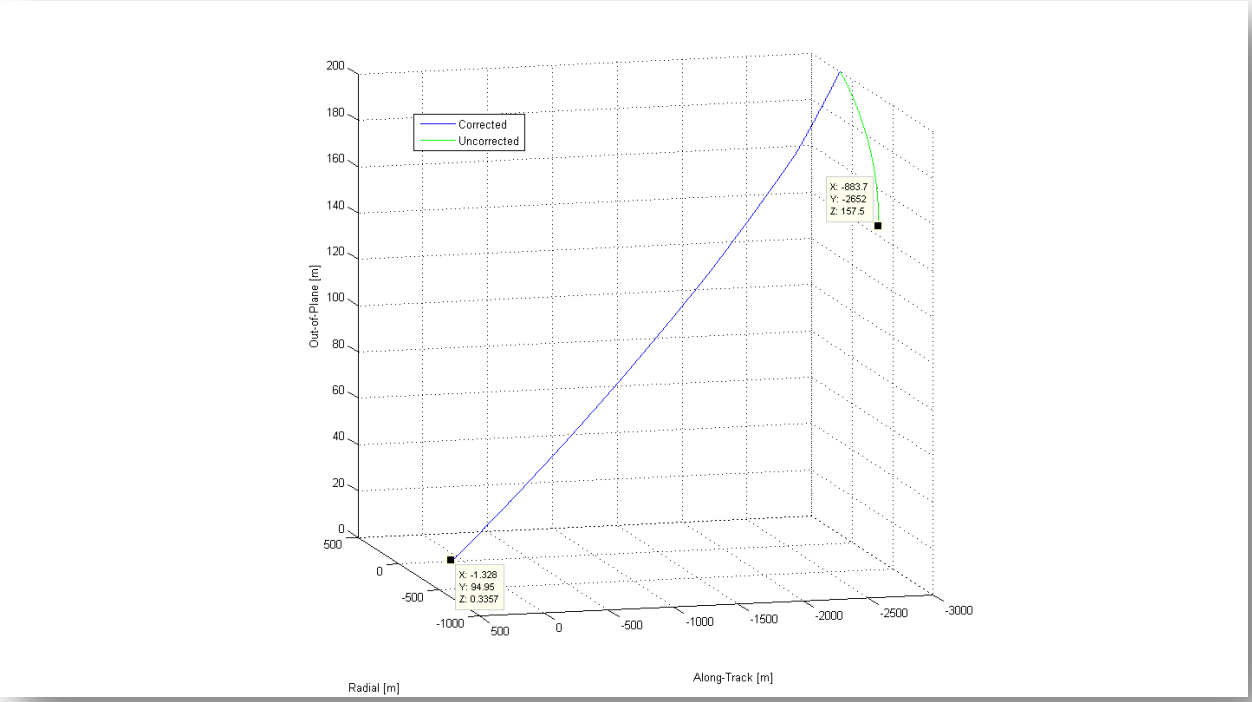

Figure 6: the initial state of the deputy (green) if it remains uncorrected.

Referring to Figure 6, the corrected (blue) line shows a final position of (-1.328, 94.95, 0.337) meters and a final velocity of (-0.001, 0.010, -0.000) meters per second is achieved. The exact *aim-point* is not achieved (0, 100, 0), due to the linear nature of the STM since the motion of the deputy is nonlinear. But it is a very fast analytical solution that achieves great results.

One possible method for reducing this error would be to use a nonlinear iterative procedure. The solution of this technique could be a first initial guess for using a full nonlinear solution. Similar techniques would be a batch processor algorithm used in orbit determination where a first guess is used to initialize an iterative process that is used with nonlinear dynamics to refine the solution. [8]

CHAPTER 6

NONLINEAR FEEDBACK CONTROL LAW

The linear state transition matrix is a great analytical approximation of computing a series of maneuvers in a nonlinear system. As an alternative to impulsive maneuvers (computed with linear equations) another approach would be to compute continuous maneuvers in a feedback control system. If the deputy (actual orbit) had the capability of performing a continuous thrust for a short period of time while monitoring its current state vector, then using this technique will allow an achievement greater results. Also, it will be shown that this type of feedback control law has asymptotic stability, meaning that over time the amount of energy needed in the thrust will asymptotically decrease with time and remain stable.

1. Derivation

To begin the analysis, the following definitions are made. Letting ***u*** be the continuous control thrust vector the relative motion to be can be defined as

$$\ddot{\boldsymbol{\rho}} = f(\boldsymbol{\rho}) + \mathbf{u} \tag{82}$$

Where the relative nonlinear equation of motion is taken from Eq. (47) and repeated here. Keep in mind that Eq. (83) works for all circular and non-circular orbits and the only assumptions made were Two-body motion and that the two orbits are in close proximity to one another compared to the radial distance of the chief orbit.

$$f(\boldsymbol{\rho}) = \begin{bmatrix} 2\dot{f}\left(\dot{y} - y\frac{\dot{r}_c}{r_c}\right) + x\dot{f}^2\left(1 + 2\frac{r_c}{p}\right) \\ -2\dot{f}\left(\dot{x} - x\frac{\dot{r}_c}{r_c}\right) + y\dot{f}^2\left(1 - \frac{r_c}{p}\right) \\ -\frac{r_c}{p}\dot{f}^2 z \end{bmatrix} \tag{83}$$

In order to be able to maneuver and change the deputy's orbit, the next thing to define is the reference relative motion. This is similar to the final *aim-point* defined earlier where it is a state vector that is desired. The delta change in the state vector is then defined as the current state minus the reference state.

$$\Delta\boldsymbol{\rho} = \boldsymbol{\rho} - \boldsymbol{\rho}_r \tag{84}$$

Choosing a positive definite scalar *Lyapunov Function* (V) that is in the form of an energy function similar to the *performance index* earlier is how one can derive an asymptotically stable control. This function should be dependent on both the position and velocity as the desire is to not only meet the reference position but also the reference velocity as well. A gain matrix ($\mathbf{K}_1$) can be added to the function to allow it to be tuned to the feedback gain on the position of the deputy. This is useful for being able to better model the thrusting on a spacecraft. For larger thrusts, the gains would be higher and vice-versa for smaller thrusts. [(5) (4)]

$$V(\Delta\boldsymbol{\rho}, \Delta\dot{\boldsymbol{\rho}}) = \frac{1}{2}\Delta\dot{\boldsymbol{\rho}}^T\Delta\dot{\boldsymbol{\rho}} + \frac{1}{2}\Delta\boldsymbol{\rho}^T\mathbf{K}_1\Delta\boldsymbol{\rho} \tag{85}$$

In order to feedback the system with the current motion of a satellite, the derivative of Eq. (85) needs to be found and substituted into Eq. (83). Simplifying the derivative and collecting terms shows [5]

$$\dot{V}\left(\Delta\boldsymbol{\rho},\Delta\dot{\boldsymbol{\rho}}\right)=\Delta\dot{\boldsymbol{\rho}}^{T}\left(f(\boldsymbol{\rho})-f(\boldsymbol{\rho}_{r})+\mathbf{u}+\mathbf{K}_{1}\Delta\boldsymbol{\rho}\right) \tag{86}$$

Eq. (86) is a scalar function that is dependent on the equations of motion and the control thrust. This function should be minimized in order to make the system asymptotically stable. Ideally Eq. (86) should be negative definite. Where it is negative in all cases, but by inspection there could be cases where it will not always be negative and in particular can be zero (e.g. $\Delta\dot{\boldsymbol{\rho}}=\mathbf{0}$). So the best that can be done, is an attempt to make Eq. (86) negative semi-definite where $V=(\Delta\boldsymbol{\rho},\Delta\dot{\boldsymbol{\rho}})\leq 0$. Eq. (87) can be used to drive the control law negative but it is not negative definite as it is not a function of the position error.

$$\dot{V}=-\Delta\dot{\boldsymbol{\rho}}^{T}\mathbf{K}_{2}\Delta\dot{\boldsymbol{\rho}} \tag{87}$$

Setting Eq. (86) equal to Eq. (87) and solving for the control thrust, the negative semi-definite nonlinear feedback control thrust is [5]

$$\mathbf{u}=-\left(f(\boldsymbol{\rho})-f(\boldsymbol{\rho}_{r})\right)-\mathbf{K}_{2}\Delta\dot{\boldsymbol{\rho}}-\mathbf{K}_{1}\Delta\boldsymbol{\rho} \tag{88}$$

Since Eq. (88) is negative semi-definite, then it cannot be proven that this control law is asymptotically stable. In order to make that proof it is necessary to find a higher order derivative of Eq. (88) when $\Delta\dot{\boldsymbol{\rho}}=\mathbf{0}$. And if that higher order derivative is an odd number and is negative definite, then asymptotic stability can be proven. The scalar derivative of Eq. (87) is

$$\dddot{V} = -2\mathbf{K}_2 \Delta\dot{\boldsymbol{\rho}}^T \Delta\ddot{\boldsymbol{\rho}} \tag{89}$$

Take the derivative again, this time with Eq. (89) will find the next odd numbered derivative. Now set the condition $\Delta\dot{\boldsymbol{\rho}} = \mathbf{0}$ to see if it is negative definite.

$$\dddot{V}(\Delta\boldsymbol{\rho}, \Delta\dot{\boldsymbol{\rho}} = 0) = -2\Delta\boldsymbol{\rho}^T \mathbf{K}_1^T \mathbf{K}_2 \mathbf{K}_1 \Delta\boldsymbol{\rho} < 0 \tag{90}$$

In this case Eq. (90) is an odd numbered derivative and it is always negative. Therefore, it can be stated that the control law in Eq. (88) is asymptotically stable. Where, over time the control thrust will drive the relative state to the reference state. There is no guarantee that it will be in the shortest amount of time but the control will asymptotically decrease reducing the amount of energy used in the thrust.

Using the example in chapter five and numerically integrating the equations of motion in Eq. (83), the new nonlinear feedback control can be tested. While integrating (after each successful step) the feedback control thrust is applied with gains of ($\boldsymbol{K}_1 = 2e^{-3}\boldsymbol{I}$ and $\boldsymbol{K}_2 = 3e^{-3}\boldsymbol{I}$) to drive the actual orbit to its reference. The following table is a sampling of the control thrust over six minutes time, keeping in mind it is really a continuous thrust.

Table 3: Relative orbit feedback control thrust.

Time [s]	Thrust Magnitude [m/s]	Thrust [m/s]
0	6.220	[-0.299, 6.200, -0.400]
36	1.160	[-0.112, 1.152, -0.075]
72	0.191	[0.033, -0.187, 0.012]
108	0.220	[0.045, -0.214, 0.014]
144	0.077	[0.018, -0.075, 0.005]
180	0.010	[0.002, -0.010, 0.001]
216	0.004	[-0.002, 0.004, -0.000]
252	0.003	[-0.001, 0.003, -0.000]
288	0.001	[-0.000, 0.001, -0.000]
324	0.000	[-0.000, 0.000, -0.000]
360	0.000	[0.000, 0.000, 0.000]

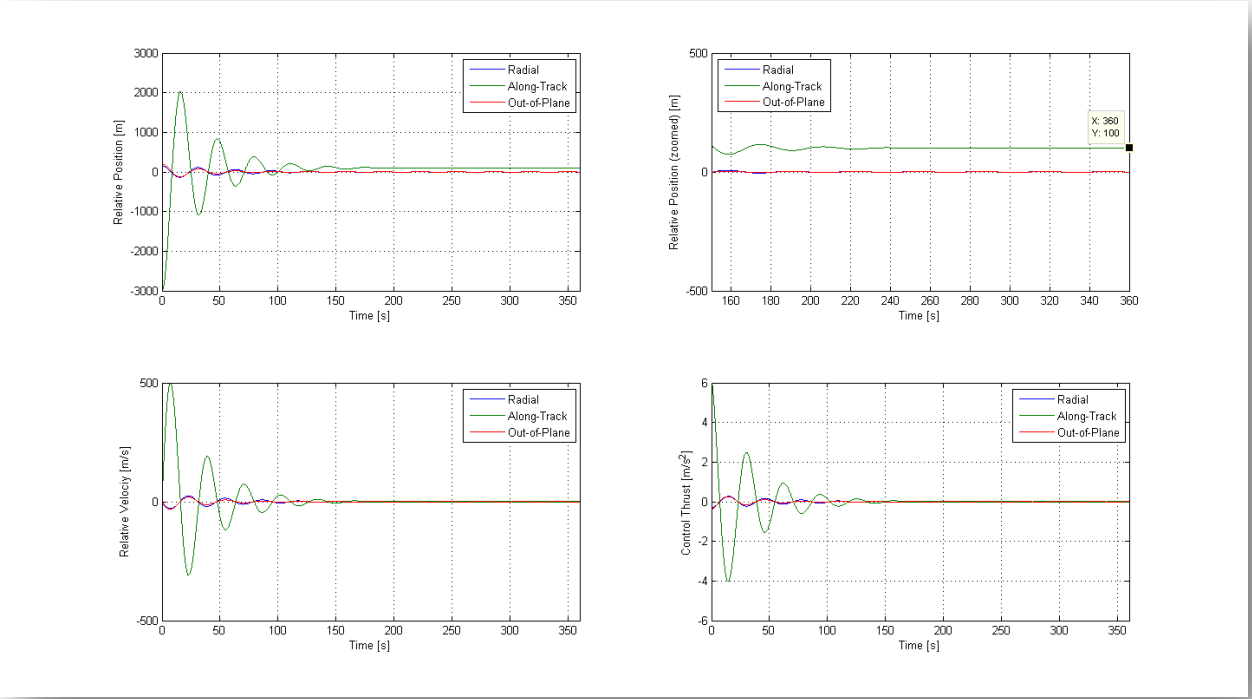

Figure 7: feedback control law applied to a geo-synchronous relative deputy orbit.

Reviewing Figure 7, the final state after applying the continuous thrust for six minutes was (-0.016, 100.033, -0.002, -0.005, 0.009, -0.001). This solution is much closer to the reference *aim-point* than the linear solution earlier. Also notice that the control is asymptotically stable.

2. Modifying the Reference Motion

Up to this point the maneuvers developed have only investigated how one might change the orbit of a spacecraft satellite relative to another. It is worth mentioning how one can change the orbit of a satellite relative to two other satellites using the previous derivations. An advantage to the previous feedback control would be to define keep-out zones (other

relative states about the chief the actual orbit should avoid) while still maintaining mission requirements. For example, if the space shuttle were to dock with the International Space Station but the shuttle must avoid a particular region of the ISS because its sensors will be in the field of view of the sun, or some debris may impact the shuttle. This could cause not only serious damage to the vehicle but to the mission and its crew as well. Instead of waiting until the sun is out of view or making guesses at the thrust direction, why not find a thrust that will push the shuttle closer to the ISS while staying away from the keep-out zone.

Using the derivations of the previous section, a second relative orbit can be defined. This second orbit is also flying relative to the chief (or ideal) orbit. It could represent some space debris or an area where the deputy (or actual) orbit must stay away from. Begin the definitions by defining the relative state of the two deputy satellites as flying relative to the chief. This definition is similar to Eq. (61)

$$\mathbf{X}_1 = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ \dot{x}_1 \\ \dot{y}_1 \\ \dot{z}_1 \end{pmatrix} \tag{91a}$$

$$\mathbf{X}_2 = \begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ \dot{x}_2 \\ \dot{y}_2 \\ \dot{z}_2 \end{pmatrix} \tag{91b}$$

If the second orbit is the orbit to stay away from, then the reference orbit can be set to the second with a sign change. This is done after each step of numerical integration when the control thrust is computed. Given some tolerance (which may be set by mission requirements), if the second relative state moves to within that tolerance to the ideal orbit then the reference motion is switched to be relative to the second. This can be done with the following

$$\boldsymbol{\rho}_r = -\boldsymbol{\rho}_2 \tag{92}$$

Once the distance between the second relative state and the ideal orbit are outside the tolerance, then the reference motion can be set back to the ideal orbit again. One way of accomplishing this task could use the following algorithm before computing the feedback control thrust. [5; 11]

$$\begin{aligned}
&d = \left|\sqrt{\boldsymbol{\rho}_2 - \boldsymbol{\rho}_r}\right| \\
&\text{if } d < \text{tolerance then} \\
&\quad \boldsymbol{\rho}_r = -\boldsymbol{\rho}_2 \\
&\text{else} \\
&\quad \boldsymbol{\rho}_r = \mathbf{0} \\
&\text{end}
\end{aligned} \tag{93}$$

$$\begin{aligned}
&\Delta\boldsymbol{\rho} = \boldsymbol{\rho}_1 - \boldsymbol{\rho}_\mathrm{r} \text{ and } \Delta\dot{\boldsymbol{\rho}} = \dot{\boldsymbol{\rho}}_1 - \dot{\boldsymbol{\rho}}_\mathrm{r} \\
&\mathbf{u} = -\left(f(\boldsymbol{\rho}_1) - f(\boldsymbol{\rho}_\mathrm{r})\right) - \mathbf{K}_2\Delta\dot{\boldsymbol{\rho}} - \mathbf{K}_1\Delta\boldsymbol{\rho}
\end{aligned}$$

Eq. (93) shows if the radial distance between the second relative position and the reference motion (in this case is zero since it is centered on the chief), is less than the tolerance; then the reference is set to the second relative orbits position with a sign change. While the ideal orbit continues to move on its path, then the actual orbit will fly relative to the second state computing a control thrust that will move the actual orbit away. Once the ideal orbit is a safe

distance away (greater than the tolerance) then the reference motion is set back to zero and the control thrust can be computed to move the actual orbit to the ideal.

Similar to the example in the previous section, the equations of motion are numerically integrated, using the previous nonlinear feedback control. Thruster gains are applied again to the control thrust of ($\boldsymbol{K}_1 = 2e^{-3}\boldsymbol{I}$ and $\boldsymbol{K}_2 = 3e^{-3}\boldsymbol{I}$). This time, for a geo-synchronous circular orbit, the following relative orbit state is used for the actual orbit.

$$\mathbf{X}_1 = \begin{pmatrix} -150 \\ 100 \\ 20 \\ 0.1 \\ -0.05 \\ -0.01 \end{pmatrix} [m, m/s]$$

The second relative orbit (that the actual orbit should stay away from) is moving about the chief at twenty meters lower radial, forty meters in-track and some velocity components.

$$\mathbf{X}_2 = \begin{pmatrix} -20 \\ 40 \\ 0 \\ 1 \\ -1 \\ 0 \end{pmatrix} [m, m/s]$$

Conducting the above example, the following table is a sampling of the control thrust over five minutes of time, keeping in mind it is really a continuous thrust.

Table 4: Relative orbit feedback control thrust with one keep-out zone.

Time [s]	Thrust Magnitude [m/s]	Thrust [m/s]
0	0.362	[0.300,-0.200, -0.040]
30	0.131	[0.118, 0.054, -0.016]

60	0.029	[0.024, -0.015, 0.006]
90	0.007	[0.004, -0.005, 0.002]
120	0.002	[-0.001, 0.002,0.000]
150	0.002	[-0.002, 0.001,0.000]
180	0.001	[-0.001,0.000,0.000]
210	0.001	[0.001,0.000,0.000]
240	0.000	[0.000,0.00,0.000]
270	0.000	[0.000,0.000, 0.000]
300	0.000	[0.000,0.000,0.000]

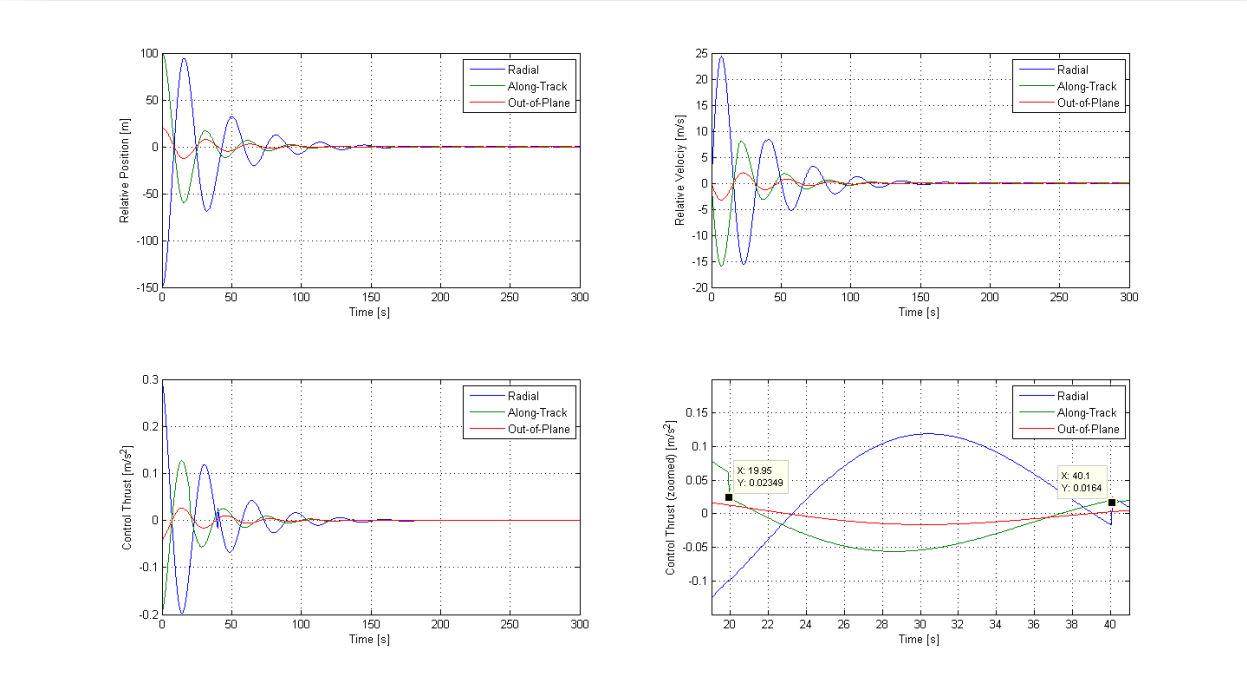

Figure 8: feedback control law applied to a geo-synchronous relative deputy orbit with a keep-out zone.

Looking at Figure 8, the final state after applying the continuous thrust for five minutes was (0.011, -0.001, -0.002, 0.003, 0.001, -0.000). As far as the relative position and velocity are concerned they settled out asymptotically, which was desired. When inspecting the control thrust more closely it becomes apparent that between twenty and forty seconds (roughly), the control changed (specifically in the radial and along-track directions). This was the expected behavior, as the second relative orbit approached the ideal, some corrective action was taken to

stay away from the second orbit until it was safe again for the control to settle out on the ideal reference motion again.

CHAPTER 7

CONCLUSIONS

1. Conclusion

Two techniques were developed that can aid in computing maneuvers in the relative rotating orbit frame. The first was a technique that leveraged on the *Clohessy-Wilshire (CW) Equations*. Using a state transition matrix, a set of impulsive maneuvers can be computed that can help attain a final orbit in a certain period of time where each maneuver is equally spaced. This technique provides the ability to quickly compute the maneuvers and achieve very accurate results. The best fit maneuver technique is only good for chief orbits that are circular and that the orbits are in close proximity to one another.

The second technique was to use a nonlinear approach by creating a feedback control law that is asymptotically stable. The advantages to this technique are the increase in accuracy and the chief orbit does not have to be circular. Implementing this control law is much more difficult as it requires numerical integration while updating the integration steps with the control thrust. The gains will have to be tuned as well depending on the thrusters on the vehicle.

Using the nonlinear approach, it was possible to modify the reference motion of a deputy orbit to modify its trajectory. Using the control law developed, earlier it was possible to set the reference motion as a function of another relative orbit. This technique has the advantage of computing a thrust that will maintain a relative orbit to the chief while avoiding any keep out zones.

2. Future Work

Due to the assumptions made in chapter three (excluding section four), all of these developments only work with Two-body motion. Since analytic solutions exist for elliptic and *J2* (earth oblateness), using classic elements [7] [3], some future work would include translating the Cartesian solutions to orbit element solutions in order to include higher fidelity information that may contain perturbations like *J2*. Using the classic orbit elements as a *6x1* vector, it is possible to obtain a direct linear mapping between the Cartesian relative orbit state and a set of orbit element differences [5].

BIBLIOGRAPHY

1. **Bate, R. R. and Mueller, D. D. and White, J. E.** *Fundamentals of Astrodynamics.* New York : Dover Publications, Inc., 1971.

2. *Fundamentals of Astrodynamics and Applications, 3rd ed.* **Vallado, D. A.** s.l. : Microcosm Press/Springer, 2007.

3. *Impulsive Spacecraft Formation Flying Control To Establish Specific Mean Orbit Elements.* **Schaub, H. and Alfriend, K. T.** 4, s.l. : Journal of Guidance, Control, and Dynamics, 2001, Vol. 24, pp. 739-745.

4. *Spacecraft Formation Flying Control using Mean Orbit Elements.* **Schaub, H. and Vadali, S. R. and Junkins, J. L. and Alfriend, K. T.** 1, s.l. : Journal of the Astronautical Sciences, 2000, Vol. 48.

5. **Schaub, H. and Junkins, J.** *Analytical Mechanics of Space Systems, 2nd ed.* s.l. : American Institute of Aeronautics and Astronautics, Inc., 2009.

6. *Terminal Guidance System for Satellite Rendezvous.* **Clohessy, W. H. and Wiltshire, R. S.** 9, s.l. : Journal of the Aerospace Sciences, 1960, Vol. 27, pp. 653-658.

7. *Dynamics and Control of Spacecraft Formations: Challenges and Some Solutions.* **Alfriend, K. T. and Schaub, H.** 2, s.l. : Journal of the Astronautical Sciences, 2000, Vol. 48, pp. 249-267.

8. **Tapley, B. D. and Schutz, B. E. and Born, G. H.** *Statistical Orbit Determination.* Burlington : Elsevier Academic Press, 2004.

9. **Berry, M. M.** A Variable-Step Double-Integration Multi-Step Integrator. s.l. : Virginia Polytechnic Institute, 2004.

10. **Montenbruck, O. and Gill, E.** *Satellite Orbits - Models, Methods and Applications.* New York : Springer, 2005.

11. *Hybrid Cartesian and Orbit Element Feedback Law for Formation Flying Spacecraft.* **Schaub, H. and Alfriend, K. T.** 2, s.l. : Journal of Guidance, Navigation and Control, 2002, Vol. 25, pp. 387-393.

12. **Borisenko, A. I. and Tarapov, I. E.** *Vector and Tensor Analysis.* New York : Dover Publications, Inc., 1968.

13. **Shampine, L. F. and Gordon, M. K.** *Computer Solution of Ordinary Differential Equations.* San Francisco : W. H. Freeman and Company, 1975.

14. **Battin, R. H.** *An Introduction to the Mathematics and Methods of Astrodynamics.* New York : AIAA Education Series, 1987.

15. **Prussing, J. E. and Conway, B. A.** *Orbital Mechanics.* New York : Oxford University Press, 1993.

16. **Ferziger, J. H.** *Numerical Methods for Engineering Application, 2nd ed.* New York : Wiley-Interscience, 1998.

17. **Kaula, W. M.** *Theory of Satellite Geodesy.* New York : Dover Publications, Inc., 2000.

18. *Implementation of Gauss-Jackson Integration for Orbit Propagation.* **Berry, M. M. and Healy L. M.** 3, s.l. : Journal of the Astronautical Sciences, 2004, Vol. 52, pp. 331-357.

19. **Trauth, M. H.** *MATLAB Recipes for Earth Sciences.* New York : Springer, 2007.

APPENDIX

The following source code was used to produce all of the results and validate the equations, including plots and example output. The software was implemented in MATLAB® by Mathworks (www.mathworks.com).

```
0001 % THESIS
0002 % $Id: thesis.m 2439 2011-05-10 03:11:51Z msaunder $
0003 %
0004 % Description:
0005 %   Execute the examples and plots.
0006 %
0007 % Prototype:
0008 %   thesis
0009 %
0010 % Inputs:
0011 %   None
0012 %
0013 % Outputs:
0014 %   None
0015 %
0016
0017 %
0018 % Copyright (C) 2011 Marc Saunders
0019 %
0020 % Licensed by Marc Saunders;
0021 % you may not use this file except in compliance with the License.
0022 %
0023 % Unless required by applicable law or agreed to in writing, software
0024 % distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
0025 % WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
0026 % License for the specific language governing permissions and limitations
0027 % under the License.
0028 %
0029
0030 function thesis
0031     clc; clear
0032     global we gmu twopi todeg torad
0033     we = 7.29211585529998e-5; % Earth's rotational rate [rad/sec].
0034     gmu = 398600.4418e9; % Earth's gravitational parameter [m^3/s^2].
0035     twopi = 2*pi; % Two-PI.
0036     todeg = 180.0 / pi; % Radians to degrees.
0037     torad = pi / 180.0; % Degrees to radians.
0038
0039     test1
0040     test2
0041     test3
0042     test4
0043     test5
0044 end
0045
0046 function test1
0047     global we
0048
0049     % Initialize the initial state and desired final state.
0050     X0 = [-50, 200, -75, 0.1, 0.01, -0.02]';
0051     Xf = [0, -100, 0, 0, 0, 0]';
0052
0053     % Get the STM.
0054     phi = stm(600, we);
```

```
0055     phi1 = phi(1:3, 1:3);
0056     phi2 = phi(1:3, 4:6);
0057     phi3 = phi(4:6, 1:3);
0058     phi4 = phi(4:6, 4:6);
0059
0060     % Compute the first maneuver and apply.
0061     dva = phi2\(Xf(1:3) - phi1*X0(1:3)) - X0(4:6);
0062     Xf1 = phi*([X0(1:3); X0(4:6) + dva]);
0063
0064     % Compute the second maneuver and apply.
0065     dvb = phi4\(Xf(4:6) - phi3*Xf1(1:3)) - Xf1(4:6);
0066     Xf1 = phi*([Xf1(1:3); Xf1(4:6) + dvb]);
0067
0068     % Output the results.
0069     fprintf('Targeting Eqn. Impulsive Maneuvers\n');
0070     fprintf('norm=%.3f, dva=[%.3f, %.3f, %.3f] (m/s)\n', norm(dva), dva);
0071     fprintf('norm=%.3f, dvb=[%.3f, %.3f, %.3f] (m/s)\n', norm(dvb), dvb);
0072     fprintf('\nTarget final state\n');
0073     fprintf('[%.3f, %.3f, %.3f, %.3f, %.3f, %.3f]\n', Xf);
0074     fprintf('\nFinal state after maneuvers\n');
0075     fprintf('[%.3f, %.3f, %.3f, %.3f, %.3f, %.3f]\n\n', Xf1);
0076 end
0077
0078 function test2
0079     global we
0080
0081     % Create a plot for a closed relative orbit solution.
0082     t = (0:1:86400)';
0083     x0 = -0.1; y0 = 0.2; z0 = -0.2; % [m]
0084     dx0 = 0.01; dy0 = 0.0; dz0 = -0.02; %[m/s]
0085     rho = soln([x0; y0; z0; dx0; dy0; dz0], t, we);
0086
0087     figure(1); clf
0088     plot3(rho(:,1), rho(:,2), rho(:,3)); grid on
0089     axis square; view(-60, 10);
0090     xlabel('Radial [m]');
0091     ylabel('Along-Track [m]');
0092     zlabel('Out-of-Plane [m]');
0093
0094     % Create a plot for a non-closed relative orbit solution.
0095     dy0 = 0.001;
0096     rho = soln([x0; y0; z0; dx0; dy0; dz0], t, we);
0097
0098     figure(2); clf
0099     plot3(rho(:,1), rho(:,2), rho(:,3)); grid on
0100     axis square; view(-60, 10);
0101     xlabel('Radial [m]');
0102     ylabel('Along-Track [m]');
0103     zlabel('Out-of-Plane [m]');
0104 end
0105
0106 function test3
0107     global we
0108
0109     % Compute a weighting matrix.
0110     num = 6;
0111     W = diag([0.1, 1, 0.1, 0.01, 1, 0.01]); % weighting matrix [x,y,z,dx,dy,dz]
0112
0113     % Compute the relative maneuvers.
0114     tf = 3600;
0115     rho0 = [150, -3000, 200, -0.3, 0.02, -0.01]'; % [m, m/s]
0116     aimpt = [0, 100, 0, 0, 0.01, 0]'; % [m, m/s]
0117     [tm, dvs] = mnvrs(num, tf, rho0, aimpt, we, W); % [s, m/s]
0118
0119     % Compute the relative orbit over time.
0120     dt = 1;
0121     t = (0:dt:tf)';
0122     phi = stm(dt, we);
0123     rho1 = zeros(length(t), 6);
```

```
0124     rho2 = zeros(length(t), 6);
0125
0126     % Loop over time.
0127     for i = 1:length(t)
0128         % Initialize.
0129         if (i == 1)
0130             prevRho1 = rho0;
0131             prevRho2 = rho0;
0132         else
0133             prevRho1 = rho1(i-1, :)';
0134             prevRho2 = rho2(i-1, :)';
0135         end
0136
0137         % Apply (impulse) the maneuver.
0138         index = find(tm == t(i));
0139         if (~isempty(index))
0140             dv = dvs(index, :);
0141             rho1(i,:) = prevRho1;
0142             rho2(i, :) = phi*prevRho2;
0143             rho1(i,4:6) = rho1(i,4:6) + dv;
0144             continue;
0145         end
0146
0147         % Propagate.
0148         rho1(i, :) = phi*prevRho1;
0149         rho2(i, :) = phi*prevRho2;
0150     end
0151
0152     % Output the results.
0153     fprintf('Relative Orbit Impulsive Maneuvers\n');
0154     for i = 1:size(dvs, 1)
0155         fprintf('%5g (s) - norm=%.3f, dv=[%.3f, %.3f, %.3f] (m/s)\n', ...
0156             tm(i), norm(dvs(i,:)), dvs(i,:));
0157     end
0158     fprintf('\nFinal state\n');
0159     fprintf('rho = [%.3f, %.3f, %.3f, %.3f, %.3f, %.3f]\n\n', rho1(end,:));
0160
0161     % Plot the paths.
0162     figure(3); clf; hold on
0163     plot3(rho1(:,1), rho1(:,2), rho1(:,3), 'b-', ...
0164           rho2(:,1), rho2(:,2), rho2(:,3), 'g-');
0165     axis square; grid on; view(-105, 10);
0166     xlabel('Radial [m]');
0167     ylabel('Along-Track [m]');
0168     zlabel('Out-of-Plane [m]');
0169     legend('Corrected', 'Uncorrected', 'Location', 'NorthWest');
0170     hold off
0171 end
0172
0173 function test4
0174     global chief K1 K2 idx rhor torad u
0175
0176     % Initialize the chief orbit.
0177     chief = [42164169.6341702, 0.001, 10*torad, 0, 0, 0]';
0178
0179     % Initial the deputy relative orbit state.
0180     X0 = [150, -3000, 200, -0.3, 0.02, -0.01]'; % [m, m/s]
0181
0182     % Initialize some values.
0183     idx = 1;
0184     K1 = 2e-3*eye(3);
0185     K2 = 3e-3*eye(3);
0186     rhor = [0, 100, 0, 0, 0.01, 0]'; % Reference orbit state, [m, m/s]
0187
0188     % Integrate the EOM.
0189     [t, X] = rk4(@test4Dydx, [0 360], X0, 0.05, @test4Step);
0190
0191     % Relative orbit results.
0192     rho  = X(:, 1:3);
```

```
0193      drho = X(:, 4:6);
0194
0195      % Output the results.
0196      fprintf('Relative Orbit Feedback Control Maneuvers\n');
0197      di = fix(0.1*size(u, 1));
0198      for i = 1:di:size(u, 1)
0199          fprintf('%5g (s) - norm=%.3f, dv=[%.3f, %.3f, %.3f] (m/s)\n', ...
0200              t(i), norm(u(i,:)), u(i,:));
0201      end
0202      fprintf('\nFinal state\n');
0203      fprintf('rho = [%.3f, %.3f, %.3f, %.3f, %.3f, %.3f]\n\n', ...
0204          rho(end,:), drho(end,:));
0205
0206      % Plot results.
0207      % Relative position.
0208      figure(4);
0209      subplot(2, 2, 1);
0210      plot(t, rho(:, 1), t, rho(:, 2), t, rho(:, 3)); grid on
0211      xlabel('Time [s]');
0212      ylabel('Relative Position [m]');
0213      legend('Radial', 'Along-Track', 'Out-of-Plane');
0214      axis([t(1) t(end) -3000 3000]);
0215
0216      % Relative position (zoomed).
0217      subplot(2, 2, 2);
0218      plot(t, rho(:, 1), t, rho(:, 2), t, rho(:, 3)); grid on
0219      xlabel('Time [s]');
0220      ylabel('Relative Position (zoomed) [m]');
0221      legend('Radial', 'Along-Track', 'Out-of-Plane', ...
0222          'Location', 'NorthWest');
0223      axis([150 t(end) -500 500]);
0224
0225      % Relative velocity.
0226      subplot(2, 2, 3);
0227      plot(t, drho(:, 1), t, drho(:, 2), t, drho(:, 3)); grid on
0228      xlabel('Time [s]');
0229      ylabel('Relative Velociy [m/s]');
0230      legend('Radial', 'Along-Track', 'Out-of-Plane');
0231      axis([t(1) t(end) -500 500]);
0232
0233      % Control thrust.
0234      subplot(2, 2, 4);
0235      plot(t, u(:, 1), t, u(:, 2), t, u(:, 3)); grid on
0236      xlabel('Time [s]');
0237      ylabel('Control Thrust [m/s^2]');
0238      legend('Radial', 'Along-Track', 'Out-of-Plane');
0239      axis([t(1) t(end) -6 6]);
0240
0241      % Clear globals
0242      clear global chief K1 K2 idx rhor u
0243  end
0244
0245  function dX = test4Dydx(t, X)
0246      global chief
0247
0248      % Get the current state.
0249      x  = X(1);
0250      y  = X(2);
0251      z  = X(3);
0252      dx = X(4);
0253      dy = X(5);
0254      dz = X(6);
0255
0256      % Update the chief orbit and convert to Cartesian.
0257      [rc, drc, f] = kep2cart(chief, t);
0258
0259      % Find some common orbit values.
0260      a = chief(1); % semi-major axis
0261      e = chief(2); % eccentricity
```

```
0262     p = a*(1 - e^2); % semi-parameter
0263     h = norm(cross(rc, drc)); % angular momentum
0264
0265     % Initialize some common values.
0266     rc = p/(1 + e*cos(f)); % orbit radius
0267     df = h/rc^2; % change in angular rate
0268     drc = (rc*e*df*sin(f))/(1 + e*cos(f)); % change in orbit radius
0269     rcp = rc/p;
0270     drc1 = drc/rc;
0271     df2 = df^2;
0272
0273     % Compute the second-order derivatives.
0274     ddx =  2*df*(dy + y*drc1) + x*df2*(1 + 2*rcp);
0275     ddy = -2*df*(dx - x*drc1) + y*df2*(1 - rcp);
0276     ddz = -rcp*df2*z;
0277
0278     % Return the derivatives.
0279     dX = [dx; dy; dz; ddx; ddy; ddz];
0280 end
0281
0282 function rho = test4Step(t, rho0)
0283     global K1 K2 idx rhor u
0284
0285     % Compute the tracking errors.
0286     delRho = rho0 - rhor;
0287
0288     % Compute the nonlinear formation flying feedback control.
0289     dX = test4Dydx(t, rho0);  frho = dX(4:6);
0290     dX = test4Dydx(t, rhor); frhor = dX(4:6);
0291     u0 = -(frho - frhor) - K2*delRho(4:6) - K1*delRho(1:3);
0292
0293     % Save the control for plotting.
0294     u(idx, :) = u0;
0295     idx = idx + 1;
0296
0297     % Return the new state.
0298     rho = rho0;
0299     rho(4:6) = rho0(4:6) + u0;
0300 end
0301
0302 function test5
0303     global chief K1 K2 idx torad u
0304
0305     % Initialize the chief orbit.
0306     chief = [42164169.6341702, 0.001, 10*torad, 0, 0, 0]';
0307
0308     % Initial the deputy's relative orbit states.
0309     X1 = [-150,  100, 20, 0.1, -0.05, -0.01]'; % [m, m/s]
0310     X2 = [-20, 40, 0, 1, -1, 0]'; % [m, m/s]
0311     Xr = [0, 0, 0, 0, 0, 0]'; % [m, m/s]
0312
0313     % Initialize some values.
0314     idx = 1;
0315     K1 = 2e-3*eye(3);
0316     K2 = 3e-3*eye(3);
0317
0318     % Integrate the EOM.
0319     [t, X] = rk4(@test5Dydx, [0 300], [X1; X2; Xr], 0.05, @test5Step);
0320
0321     % Relative orbit results.
0322     rho1  = X(:, 1:3);
0323     drho1 = X(:, 4:6);
0324
0325     % Output the results.
0326     fprintf('Relative Orbit Feedback Control Maneuvers\n');
0327     di = fix(0.1*size(u, 1));
0328     for i = 1:di:size(u, 1)
0329         fprintf('%5g (s) - norm=%.3f, dv=[%.3f, %.3f, %.3f] (m/s)\n', ...
0330             t(i), norm(u(i,:)), u(i,:));
```

```
0331      end
0332      fprintf('\nFinal state\n');
0333      fprintf('rho = [%.3f, %.3f, %.3f, %.3f, %.3f, %.3f]\n\n', ...
0334          rho1(end,:), drho1(end,:));
0335
0336      % Plot results.
0337      % Relative position.
0338      figure(5);
0339      subplot(2, 2, 1);
0340      plot(t, rho1(:, 1), t, rho1(:, 2), t, rho1(:, 3)); grid on
0341      xlabel('Time [s]');
0342      ylabel('Relative Position [m]');
0343      legend('Radial', 'Along-Track', 'Out-of-Plane');
0344
0345      % Relative velocity.
0346      subplot(2, 2, 2);
0347      plot(t, drho1(:, 1), t, drho1(:, 2), t, drho1(:, 3)); grid on
0348      xlabel('Time [s]');
0349      ylabel('Relative Velociy [m/s]');
0350      legend('Radial', 'Along-Track', 'Out-of-Plane');
0351
0352      % Control thrust.
0353      subplot(2, 2, 3);
0354      plot(t, u(:, 1), t, u(:, 2), t, u(:, 3)); grid on
0355      xlabel('Time [s]');
0356      ylabel('Control Thrust [m/s^2]');
0357      legend('Radial', 'Along-Track', 'Out-of-Plane');
0358
0359      % Relative position (zoomed).
0360      subplot(2, 2, 4);
0361      plot(t, u(:, 1), t, u(:, 2), t, u(:, 3)); grid on
0362      xlabel('Time [s]');
0363      ylabel('Control Thrust (zoomed) [m/s^2]');
0364      legend('Radial', 'Along-Track', 'Out-of-Plane');
0365      axis([19 41 -0.15 0.2]);
0366
0367      % Clear globals
0368      clear global chief K1 K2 idx rhor u
0369  end
0370
0371  function dX = test5Dydx(t, X)
0372      global chief
0373
0374      % Get the current state.
0375      x1  = X(1);
0376      y1  = X(2);
0377      z1  = X(3);
0378      dx1 = X(4);
0379      dy1 = X(5);
0380      dz1 = X(6);
0381
0382      x2  = X(7);
0383      y2  = X(8);
0384      z2  = X(9);
0385      dx2 = X(10);
0386      dy2 = X(11);
0387      dz2 = X(12);
0388
0389      xr  = X(13);
0390      yr  = X(14);
0391      zr  = X(15);
0392      dxr = X(16);
0393      dyr = X(17);
0394      dzr = X(18);
0395
0396      % Update the chief orbit and convert to Cartesian.
0397      [rc, drc, f] = kep2cart(chief, t);
0398
0399      % Find some common orbit values.
```

```
0400      a = chief(1); % semi-major axis
0401      e = chief(2); % eccentricity
0402      p = a*(1 - e^2); % semi-parameter
0403      h = norm(cross(rc, drc)); % angular momentum
0404
0405      % Initialize some common values.
0406      rc = p/(1 + e*cos(f)); % orbit radius
0407      df = h/rc^2; % change in angular rate
0408      drc = (rc*e*df*sin(f))/(1 + e*cos(f)); % change in orbit radius
0409      rcp = rc/p;
0410      drc1 = drc/rc;
0411      df2 = df^2;
0412
0413      % Compute the second-order derivative for X1.
0414      ddx1 =  2*df*(dy1 + y1*drc1) + x1*df2*(1 + 2*rcp);
0415      ddy1 = -2*df*(dx1 - x1*drc1) + y1*df2*(1 - rcp);
0416      ddz1 = -rcp*df2*z1;
0417
0418      % Compute the second-order derivative for X2.
0419      ddx2 =  2*df*(dy2 + y2*drc1) + x2*df2*(1 + 2*rcp);
0420      ddy2 = -2*df*(dx2 - x2*drc1) + y2*df2*(1 - rcp);
0421      ddz2 = -rcp*df2*z2;
0422
0423      % Compute the second-order derivative for Xr.
0424      ddxr =  2*df*(dyr + yr*drc1) + xr*df2*(1 + 2*rcp);
0425      ddyr = -2*df*(dxr - xr*drc1) + yr*df2*(1 - rcp);
0426      ddzr = -rcp*df2*zr;
0427
0428      % Return the derivatives.
0429      dX1 = [dx1; dy1; dz1; ddx1; ddy1; ddz1];
0430      dX2 = [dx2; dy2; dz2; ddx2; ddy2; ddz2];
0431      dXr = [dxr; dyr; dzr; ddxr; ddyr; ddzr];
0432      dX  = [dX1; dX2; dXr];
0433  end
0434
0435  function X = test5Step(t, X0)
0436      global K1 K2 idx u
0437
0438      % Get the current state.
0439      rho1  = X0(1:3);
0440      drho1 = X0(4:6);
0441      rho2  = X0(7:9);
0442      drho2 = X0(10:12);
0443      rhor  = X0(13:15);
0444      drhor = X0(16:18);
0445
0446      % Compute the equations of motion.
0447      dX = test5Dydx(t, X0);
0448      frho1 = dX(4:6);
0449      frho2 = dX(10:12);
0450      frhor = dX(13:15);
0451
0452      % Compute the distance between the reference motion and rho2.
0453      d = norm(rho2 - rhor);
0454      if (d < 20)
0455          ref  = -rho2;
0456          dref = -drho2;
0457          frho = -frho2;
0458      else
0459          ref  = rhor;
0460          dref = drhor;
0461          frho = frhor;
0462      end
0463
0464      % Compute the tracking error.
0465      delRho = rho1 - ref;
0466      delRhoDot = drho1 - dref;
0467
0468      % Compute the nonlinear formation flying feedback control.
```

```
0469     u0 = -(frho1 - frho) - K2*delRhoDot - K1*delRho;
0470
0471     % Save the control for plotting.
0472     u(idx, :) = u0;
0473     idx = idx + 1;
0474
0475     % Return the new state.
0476     X = X0;
0477     X(4:6) = X0(4:6) + u0;
0478 end
0479
0480 % Convert classic Kepler orbit elements to Cartesian vectors.
0481 function [r, dr, f] = kep2cart(orb, dt)
0482     % Declare globals.
0483     global gmu twopi
0484
0485     % Set dt to zero if it is not given.
0486     if (nargin == 1)
0487         dt = 0;
0488     end
0489
0490     % Set the orbit elements.
0491     a = orb(1);
0492     e = orb(2);
0493     i = orb(3);
0494     O = orb(4);
0495     w = orb(5);
0496     p = a * (1 - e^2); % Semi-parameter
0497     f = mean2tru(orb, dt); % Find the tru anomaly.
0498
0499     % Correct the orbit plane if we are an elliptical orbit for circular
0500     % and equitorial cases.
0501     tol = 1e-8;
0502     if (a > 0)
0503         if (e < tol)
0504             if (i < tol)
0505                 % Circular and equitorial.
0506                 f = xmod(O + w + f, twopi);
0507                 w = 0;
0508                 O = 0;
0509             else
0510                 % Circular and inclined.
0511                 f = xmod(w + f, twopi);
0512                 w = 0;
0513             end
0514         else
0515             % Elliptical and equatorial.
0516             if (i < tol)
0517                 w = xmod(O + w, twopi);
0518                 O = 0;
0519             end
0520         end
0521     end
0522
0523     % Pre-compute some common values.
0524     cosO = cos(O);
0525     sinO = sin(O);
0526     cosW = cos(w);
0527     sinW = sin(w);
0528     cosI = cos(i);
0529     sinI = sin(i);
0530     cosF = cos(f);
0531     sinF = sin(f);
0532
0533     % Perifocal system unit vectors.
0534     phat = [
0535         cosO*cosW - sinO*sinW*cosI, ...
0536         sinO*cosW + cosO*sinW*cosI, ...
0537         sinW*sinI ...
```

```
0538     ]';
0539
0540     qhat = [
0541         -cosO*sinW - sinO*cosW*cosI, ...
0542         -sinO*sinW + cosO*cosW*cosI, ...
0543          cosW*sinI ...
0544     ]';
0545
0546     % Position magnitude.
0547     rmag = p / (1 + e*cosF);
0548
0549     % Position and velocity vectors.
0550     r  = ((rmag*cosF) * phat) + ((rmag*sinF) * qhat);
0551     dr = sqrt(gmu / p) * (-sinF*phat + (e + cosF)*qhat);
0552 end
0553
0554 % Given classic orbit elements with mean anomaly and an optional change in
0555 % time, return the true anomaly.
0556 function f = mean2tru(orb, dt)
0557     % Declare globals.
0558     global gmu twopi
0559
0560     % Set dt to zero if it is not given.
0561     if (nargin == 1)
0562         dt = 0;
0563     end
0564
0565     % Get the orbit elements.
0566     a  = orb(1);
0567     e  = orb(2);
0568     M0 = orb(6);
0569
0570     % Elliptic
0571     % Propagate the orbit (without disturbances) if dt is not zero.
0572     M = M0;
0573     if (dt ~= 0)
0574         n = sqrt(gmu / a^3); % Mean angular motion
0575         M = M0 + n*dt;
0576     end
0577
0578     % Use a Newton root finder to compute anomaly.
0579     E = M;
0580     dE = 1e10;
0581     while (abs(dE) > 10*eps)
0582         dE = (E - e * sin(E) - M) / (1 - e * cos(E));
0583         E  = E - dE;
0584     end
0585
0586     % True anomaly.
0587     tanf2 = sqrt((1 + e) / (1 - e)) * tan(E / 2);
0588     f = xmod(2 * atan(tanf2), twopi);
0589 end
0590
0591 % Return maneuvers.
0592 function [t, dvs] = mnvrs(num, tf, rho0, aimpt, n, W)
0593     % Handle the input arguments.
0594     if (nargin < 6)
0595         W = eye(length(rho0));
0596     end
0597
0598     % Compute the time of each dv.
0599     dt = tf/num;
0600     t = (0:dt:tf)';
0601     t = t(1:end-1);
0602
0603     % Compute the error between the current state and the final aim-point.
0604     phi = stm(tf, n);
0605     err = aimpt - phi*rho0;
0606
```

```
0607     % Loop through all the points.
0608     j = 1;
0609     psi = zeros(length(rho0), 3*num);
0610     for i = 0:num-1
0611         phi = stm(tf - i*dt, n);
0612         psi(:, j:j+2) = phi(:, 4:6);
0613         j = j + 3;
0614     end
0615
0616     % Solve the weighted least squares problem to find the DV's.
0617     H = psi;
0618     P = pinv(H'*W*H);
0619     dvs = P*H'*W*err;
0620
0621     % Reshape the matrix into the dvs.
0622     m = size(dvs, 1);
0623     dvs = reshape(dvs, 3, m/3)';
0624 end
0625
0626 % Numerically integrate using the Fourth-order Runge-Kutta method.
0627 function [t, y] = rk4(odefun, tspan, y0, h, stepfun)
0628     % Initialize the time steps.
0629     t0 = tspan(1);
0630     tf = tspan(2);
0631     dt = tf - t0;
0632
0633     % Set a default step-size if one was not given.
0634     if (nargin == 3)
0635         h = 1e-3;
0636     end
0637
0638     h = abs(h) * sign(dt);
0639     hhalf = 0.5 * h;
0640
0641     % Compute the coefficients.
0642     a = [sqrt(0.5) - 0.5, 1 - sqrt(0.5), 1 / sqrt(2), 1 + sqrt(0.5)]';
0643     b = [1/6, 1/3*(1 - sqrt(0.5)), 1/3*(1 + sqrt(0.5)), 1/6]';
0644
0645     % Initialize the integration.
0646     i = 1;
0647     n = abs(fix(dt / h)) + 1;
0648     t = zeros(n, 1);
0649     y = zeros(n, length(y0));
0650     ytmp = y0;
0651
0652     % Loop until the end of the time span.
0653     for ttmp = t0:h:tf
0654         % Store the output values.
0655         t(i) = ttmp;
0656         y(i, :) = ytmp;
0657         i = i + 1;
0658
0659         % Compute the weighted averages.
0660         k1 = h * feval(odefun, ttmp, ytmp);
0661         k2 = h * feval(odefun, ttmp + hhalf, ytmp + k1/2);
0662         k3 = h * feval(odefun, ttmp + hhalf, ytmp + a(1)*k1 + a(2)*k2);
0663         k4 = h * feval(odefun, ttmp + h, ytmp - a(3)*k2 + a(4)*k3);
0664
0665         % Compute the next value.
0666         ytmp = ytmp + (b(1)*k1 + b(2)*k2 + b(3)*k3 + b(4)*k4);
0667
0668         % Call the step function (if given).
0669         if (nargin == 5)
0670             ytmp = feval(stepfun, ttmp, ytmp);
0671         end
0672     end
0673 end
0674
0675 % Returns the solution of the two-body relative orbit.
```

```
0676 function rho = soln(rho0, t, n)
0677     x0 = rho0(1);
0678     y0 = rho0(2);
0679     z0 = rho0(3);
0680     dx0 = rho0(4);
0681     dy0 = rho0(5);
0682     dz0 = rho0(6);
0683
0684     nt = n*t;
0685     cosnt = cos(nt);
0686     sinnt = sin(nt);
0687     ninv = 1/n;
0688
0689     x = (4 - 3*cosnt)*x0 + dx0*ninv*sinnt + 2*ninv*(1 - cosnt)*dy0;
0690     y = 6*(sinnt - nt)*x0 + y0 + 2*ninv*(cosnt - 1)*dx0 + ...
0691         (-3*t + 4*ninv*sinnt)*dy0;
0692     z = cosnt*z0 + dz0*ninv*sinnt;
0693     dx = 3*n*sinnt*x0 + cosnt*dx0 + 2*sinnt*dy0;
0694     dy = 6*n*(cos(nt) - 1)*x0 - 2*sinnt*dx0 + (4*cosnt - 3)*dy0;
0695     dz = -n*sin(nt)*z0 + cosnt*dz0;
0696
0697     rho = [x, y, z, dx, dy, dz];
0698 end
0699
0700 % Returns the state transition matrix.
0701 function phi = stm(t, n)
0702     % Initialize some common values.
0703     nt = n*t;
0704     cosnt = cos(nt);
0705     sinnt = sin(nt);
0706     ninv = 1/n;
0707
0708     % Set STM values.
0709     % phi11
0710     phi = zeros(6, 6);
0711     phi(1, 1) = 4 - 3*cosnt;
0712     phi(2, 1) = 6*(sinnt - nt);
0713     phi(2, 2) = 1;
0714     phi(3, 3) = cosnt;
0715
0716     % phi12
0717     phi(1, 4) = ninv*sinnt;
0718     phi(1, 5) = 2*ninv*(1 - cosnt);
0719     phi(2, 4) = 2*ninv*(cosnt - 1);
0720     phi(2, 5) = 4*ninv*sinnt - 3*t;
0721     phi(3, 6) = ninv*sinnt;
0722
0723     % phi21
0724     phi(4, 1) = 3*n*sinnt;
0725     phi(5, 1) = 6*n*(cosnt - 1);
0726     phi(6, 3) = -n*sinnt;
0727
0728     % phi22
0729     phi(4, 4) = cosnt;
0730     phi(4, 5) = 2*sinnt;
0731     phi(5, 4) = -2*sinnt;
0732     phi(5, 5) = 4*cosnt - 3;
0733     phi(6, 6) = cosnt;
0734 end
0735
0736 % Returns the extended modulo of two values.
0737 function z = xmod(x, y)
0738     if (y == 0)
0739         z = x;
0740         return;
0741     end
0742
0743     z = x - y .* floor((x./y) + 0.5);
0744 end
```